'68' MICRO JOURNAL

$2.95 USA

Australia	A $ 4.75	New Zealand	NZ $ 6.50
Singapore	S $ 9.45	Hong Kong	H $23.50
Malaysia	M $ 9.45	Sweden	30.-SEK

VOLUME VII ISSUE III • Devoted to the 68XX User • March 1985

"Small Computers Doing Big Things"

PHOTO CREDIT: NASA

WE DON'T PLAY GAMES

X-12+ A SERIOUS COMPUTER IN A DESKTOP PACKAGE

Multiprocessor Technology - Combination of 8, 16 and 32 bit types

1.0 Megabyte Memory - Insures no limitation on programs

"Winchester" Disk System - Fast response, large storage capacity

UniFlex* Operating System - The standard of comparison

Hardware Floating Point - Unmatched speed in a small system

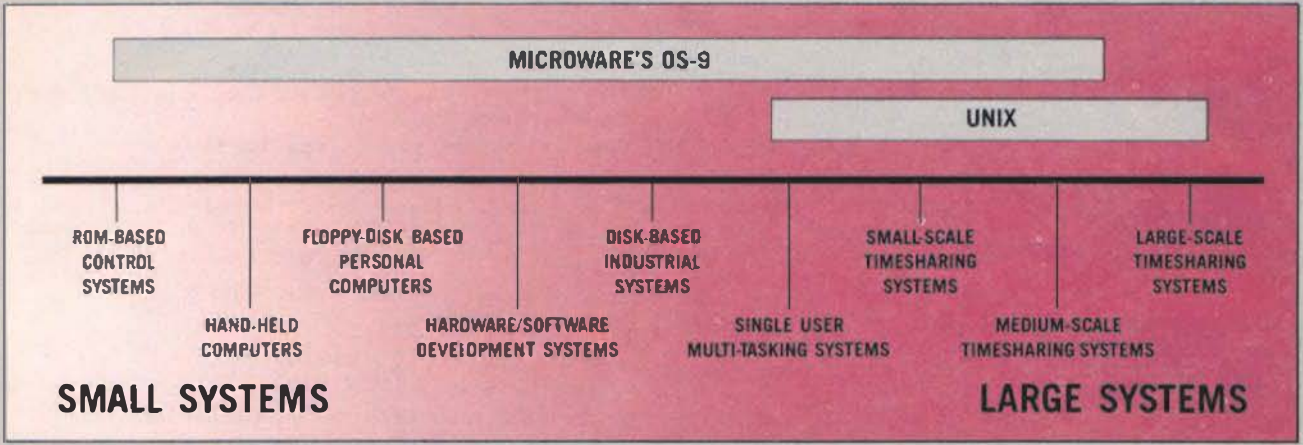
Up to Three Terminals - Instant expansion

*Trademark of Technical Systems Consultants

SWTPc

SOUTHWEST TECHNICAL PRODUCTS CORPORATION
219 W. RHAPSODY
SAN ANTONIO, TEXAS 78216
(512) 344-0241

Only Microware's OS-9 Operating System Covers the Entire 68000 Spectrum


MICROWARE'S OS-9

UNIX

ROM-BASED CONTROL SYSTEMS

HAND-HELD COMPUTERS

FLOPPY-DISK BASED PERSONAL COMPUTERS

HARDWARE/SOFTWARE DEVELOPMENT SYSTEMS

DISK-BASED INDUSTRIAL SYSTEMS

SINGLE USER MULTI-TASKING SYSTEMS

SMALL-SCALE TIMESHARING SYSTEMS

MEDIUM-SCALE TIMESHARING SYSTEMS

LARGE-SCALE TIMESHARING SYSTEMS

SMALL SYSTEMS

LARGE SYSTEMS

Is complicated software and expensive hardware keeping you back from Unix? Look into OS-9, the operating system from Microware that gives 68000 systems a Unix-style environment with much less overhead and complexity.

OS-9 is versatile, inexpensive, and delivers outstanding performance on any size system. The OS-9 executive is much smaller and far more efficient than Unix because it's written in fast, compact assembly language, making it ideal for critical real-time applications. OS-9 can run on a broad range of 8 to 32 bit systems based on the 68000 or 6809 family MPUs from ROM-based industrial controllers up to large multiuser systems.

Key OS-9 Features At A Glance

- Compact (16K) ROMable executive written in assembly language
- User "shell" and complete utility set written in C
- C-source code level compatibility with Unix
- Full Multitasking/multiuser capabilities
- Modular design - extremely easy to adapt, modify, or expand
- Unix-type tree structured file system
- Rugged "crash-proof" file structure with record locking
- Works well with floppy disk or ROM-based systems
- Uses hardware or software memory management
- High performance C, Pascal, Basic and Cobol compilers

OS-9'S OUTSTANDING C COMPILER IS YOUR BRIDGE TO UNIX

Microware's C compiler technology is another OS-9 advantage. The compiler produces extremely fast, compact, and ROMable code. You can easily develop and port system or application software back and forth to standard Unix systems. Cross-compiler versions for VAX and PDP-11 make coordinated Unix/OS-9 software development a pleasure.

SUPPORT FOR MODULAR SOFTWARE — AN OS-9 EXCLUSIVE

Comprehensive support for modular software puts OS-9 a generation ahead of other operating systems. It multiplies programmer productivity and memory efficiency. Application software can be built from individually testable software modules including standard "library" modules. The modular structure lets you customize and reconfigure OS-9 for specific hardware easily and quickly.

A SYSTEM WITH A PROVEN TRACK RECORD

Once an underground classic, OS-9 is now a solid hit. Since 1980 OS-9 has been ported to over a hundred 6809 and 68000 systems under license to some of the biggest names in the business. OS-9 has been imbedded in numerous consumer, industrial, and OEM products, and is supported by many independent software suppliers.

microware®
OS-9™

MICROWARE SYSTEMS CORPORATION
1866 NW 114th Street
Des Moines, Iowa 50322
Phone 515-224-1929
Telex 910-520-2535

Microware Japan, Ltd
3-8-9 Baraki, Ichikawa City
Chiba 272-01, Japan
Phone 0473(28)4493
Telex 299-3122

OS-9 is a trademark of Microware and Motorola. Unix is a trademark of Bell Labs.

'68' MICRO JOURNAL

Portions of the text for 68 MICRO JOURNAL was prepared using the following furnished hard/software.

COMPUTERS-HARDWARE
Southwest Technical Products
219 W. Rhapsody
San Antonio, Texas 78216
SO9-5/8 DMF disk-CDS1-8212W-Sprint 3 Printer
-
GIMIX Inc.
1337 West 37th Place
Chicago, IL 60609
Super Mainframe-OS9-FLEX-Assorted Hardware

EDITORS-WORD PROCESSORS
Technical Systems Consultants, Inc.
111 Providence Road
Chapel Hill, NC 27514
FLEX-Editor-Processor
-
Great Plains Computer ompany, Inc.
PO Box 916
Idaho Falls, ID 83401
STYLO-Mail Merge

Editorial Staff

Don Williams Sr.	Publisher
Larry E. Williams	Executive Editor
Tom E. Williams	Production Editor
Robert (Bob) Nay	olor Editor

Administrative Staff

Mary Robertson	Office Manager
Penny Williams	Subscriptions
Michael Westfall	Shipping/Rec.
Christine Kocher	Accounting

Contributing Editors

Ron Anderson
Norm Commo
Peter Dibble
Dr. Theo Elbert
William E. Fisher
Dr. E.M. Pass

Special Technical Projects

Clay Abrams K6AEP
Tom Hunt

CONTENTS

Vol.VII,Issue III March 85

Send All Correspondence To:

Computer Publishing Center
68 MICRO JOURNAL
5900 Cassandra Smith
PO Box 849
Hixson, TN 37343
Ph (615)842-4600 TELEX 558 414 PVT BTH

68' Micro Journal is published 12 times a year by Computer Publishing Inc. Second Class Postage Paid ISSN 0194-5025 at Hixson, Tenn. and additional entries. Postmaster: send Form 3579 to 68' Micro Journal, PO Box 849, Hixson, Tennessee.

SUBSCRIPTION RATES
USA
1-Year $24.50 2-Years $42.50 3-Years $64.50
FOREIGN
See Page 60

Items Submitted for Publication

Articles submitted for publication should be accompanied by the authors **full name, address, date and telephone number.** It is preferred that articles be submitted on either 5 or 8 inch diskette in TSC Editor format or STYLO format. All diskettes will ba returned.

The following TSC Text Processor commands ONLY should be used (due to our proportional processor): .sp space, .pp paragraph, .fi fill and .nf no fill. Also please do not format within the text with multiple spaces. The rest we will enter at time of editing.

STYLO commands are all acceptable except the .pg page command, we print edited text files in continous text.

All articles submitted on diskettes should be in TSC FLEX* format, either FLEX2 6800, or FLEX9 6809 any version.

If articles are submitted on paper they should be on white 8X11 bond or better grade paper. No hand written articles (hand written or drawn art accepted). All paper submitted articles will be photo reproduced. This requires that they be typed or produced with a dark ribbon (no blue), single spaced and type font no smaller than 'elite' or 12 pitch. Typed text should be approximately 7 inches wide (will be reduced to column width of 3 1/2 inches). **Please use a dark ribbon!**

All letters to the editor should also comply with the above and bear a signature. Letters of 'gripes' as well as 'praise' are solicited. We attempt to publish all letters to the editor verbatim, however, we reserve the right to reject any submission for lack of 'good taste'. We reserve the right to define what constitutes 'good taste'.

Advertising: Commercial advertisers please contact the 68 Micro Journal advertising department for current rate sheet and requirements.

Classified: All classified must be non-commercial. Maximum 20 words per classified ad. Those consisting of more than 20 words should be figured at .35 cents per word. 20 words or less $7.50 minimum, one time, paid in advance. No classified ads accepted by telephone.

GIMIX HAS THE 6809 SYSTEM TO SUIT YOUR NEEDS

HARDWARE

All systems feature the **GIMIX CLASSY CHASSIS**: with a ferro-resonant constant voltage power supply, gold plated bus connectors, and plenty of capacity for future expansion.

Static **RAM** and double-density **DMA** floppy disk controllers are used exclusively in all systems.

All systems are guaranteed for 2 MHz operation and include complete hardware and software documentation, necessary cables, filler plates, etc.

Systems are assembled using burned-in and tested boards, and all disk drives are tested and aligned by **GIMIX**.

You can add additional components to any system when ordering, or expand it in the future by adding **RAM**, **I/O**, etc.

GIMIX lets you choose from a wide variety of options to customize your system to your needs.

SOFTWARE

All **OS-9/FLEX** systems allow you to software select either operating system.

Also included is the **GMXBUG** monitor and, in systems with 128K or more of RAM, **GMX-VDISK** for **FLEX**.

All **GIMIX OS-9** systems include **Microware's Editor, Assembler, Debugger, Basic09,** and **Runb**; and the GMX versions of **RMS** and **DO** for **OS-9**.

All **GIMIX** versions of **OS-9** can read and write RS color computer format OS-9 disks, as well as the Microware/GIMIX standard format.

New and exclusive with **OS-9 GMX III** systems is the **GMX OS-9 Support ROM**, a monitor for **OS-9** that includes memory diagnostics and allows the system to boot directly from either hard disk or floppy.

A wide variety of languages and other software is available for use with either **OS-9** or **FLEX**.

OS-9 GMX III/FLEX SYSTEMS (#79)

The #79 super system now includes (in addition to the above): the **GMX 6809 CPU III**, a **256K CMOS Static RAM Board (#72)**, and a **3-port Intelligent Serial I/O Processor (#11)**.

The **GMX 6809 CPU III** can perform high-speed DMA transfers from memory to memory and uses memory attributes and illegal instruction trapping to protect the system and users from program crashes. If a user program crashes, only that user is affected; other users are unaware of the problem.

The 3-Port Intelligent Serial I/O Board (#11) significantly reduces system overhead by handling routine I/O functions; freeing the host CPU for running user programs. This improves overall system performance and allows user terminals to be run at up to 19.2K baud.

with dual 40 track DSDD drives	**$5998.79**
with dual 80 track OSDD drives	**$6198.79**
with **#88** dual 8'' DSDD drive system	**$7698.79**
with **#90** 19MB Winchester subsystem and one 80 track	**$8898.79**
with a 47MB Winchester subsystem and one 80 track	**$10,898.79**
with a 47MB plus a 6MB removable pack Winchester subsystem and one 80 track drive	**$12,398.79**

TO ORDER BY MAIL: SEND CHECK OR MONEY ORDER OR USE YOUR VISA OR MASTER CHARGE. Please allow 3 weeks for personal checks to clear. U.S. orders add $5 handling if order is under $200.00. Foreign orders add $10 handling if order is under $200.00. Foreign orders over $200.00 will be shipped via Emery Air Freight COLLECT, and we will charge no handling. All orders must be prepaid in U.S. funds. Please note that foreign checks have been taking about 8 weeks for collection so we would advise wiring money, or checks drawn on a bank account in the U.S. Our bank is the Continental Illinois National Bank of Chicago, 231 S. LaSalle Street, Chicago, IL 60693, account #73-32033.

BASIC-09 and OS-9 are trademarks of Microware Systems Corp. and MOTOROLA, Inc.
FLEX and UniFLEX are trademarks of Technical Systems Consultants, Inc.
GIMIX, GHOST, GMX, CLASSY CHASSIS, are trademarks of GIMIX, Inc.

OS-9 GMX I / FLEX SYSTEMS #49

The #49 systems include 64KB static RAM, #05 CPU, #43 2 port serial board.

with dual 40 track DSDD drives	**$3998.49**
with dual 80 track DSDD drives	**$4198.49**
with **#88** dual 8'' DSDD drive system	**$5698.49**
with **#90** 19MB Winchester subsystem and one 80 track	**$6898.49**

OS-9 GMX II / FLEX SYSTEMS #39

The #39 systems include 128KB static RAM, #05 CPU, #43 2 port serial board.

with dual 40 track DSDD drives	**$4498.39**
with dual 80 track DSDD drives	**$4698.39**
with **#88** dual 8'' DSDD drive system	**$6198.39**
with **#90** 19MB Winchester subsystem and one 80 track	**$7398.39**

GIMIX DOES NOT GUARANTEE PERFORMANCE OF ANY GIMIX SYSTEMS, BOARDS OR SOFTWARE WHEN USED WITH OTHER MANUFACTURERS PRODUCT.

EXPORT MODELS: ADD $30 FOR 50Hz. POWER SUPPLIES.

GIMIX, Inc. reserves the right to change pricing, terms, and products specifications at any time without further notice.

ALL PRICES ARE F.O.B. CHICAGO

Contact GIMIX for price and availability of UniFLEX and UniFLEX GMXIII Systems.

NOTE on all drive systems: Dual 40 track drives have about 700KB of formatted capacity; dual 80's about 1,400KB; dual 8'' about 2,000KB. The formatted capacity of hard disks is about 80% of the total capacity.

Want to expand your system to a megabyte of Static RAM and 15 users?

Simply add additional memory and I/O boards. Your **GIMIX** system can grow with your needs. Contact us for a complete list of available boards and options.

#72 256KB CMOS STATIC RAM board with battery back up	**$1898.72**
#64 64KB CMOS STATIC RAM board with battery back up	**$528.64**
#67 64KB STATIC RAM board	**$478.67**
#11 3 port intelligent serial I/O board	**$498.11**
#43 2 port serial I/O board	**$128.43**
#42 2 port parallel I/O board	**$88.42**
#95 cable sets (1 needed per port), specify board	**$24.95**

NOW SHIPPING !

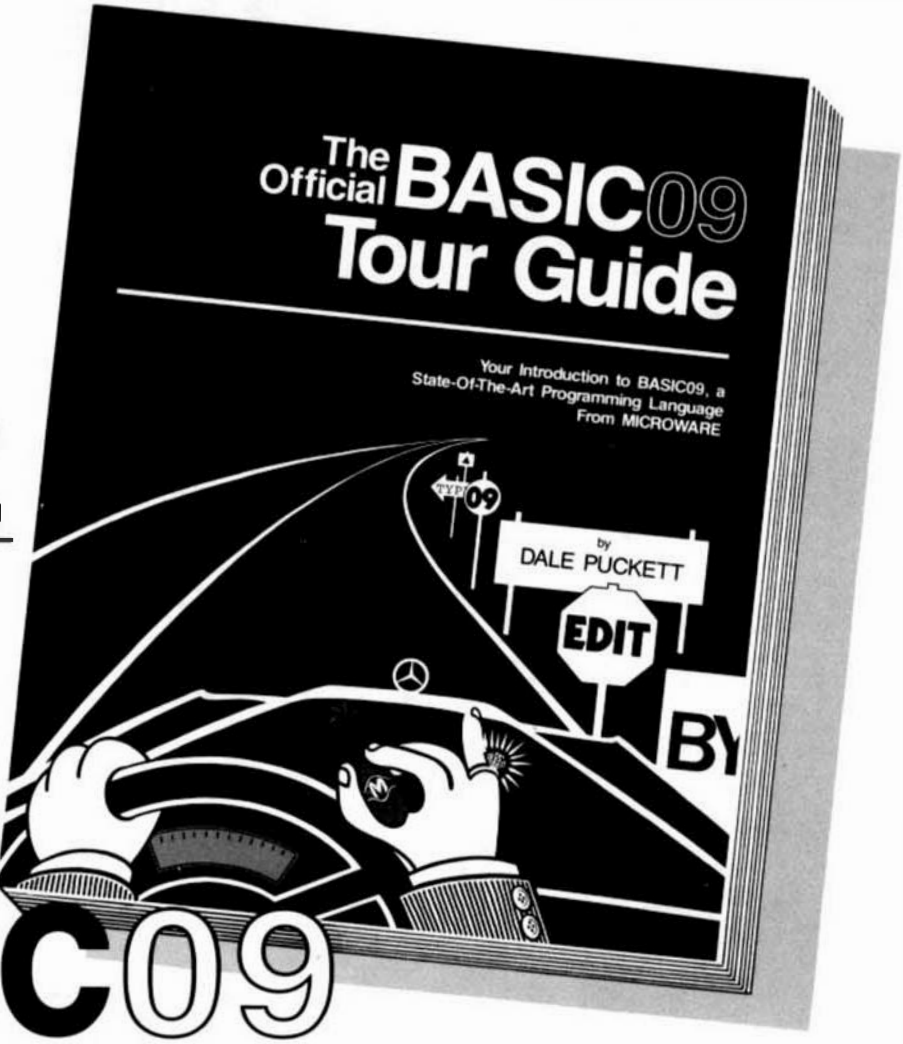
UniFLEX

GMX III Systems

GIMIX Inc.

1337 WEST 37th PLACE
CHICAGO, ILLINOIS 60609
(312) 927-5510 • TWX 910-221-4055

Get the most out of BASIC09


The Official BASIC09 Tour Guide
Your Introduction to BASIC09, a State-Of-The-Art Programming Language From MICROWARE
TYPE
09
by DALE PUCKETT
EDIT
BY

The **OFFICIAL BASIC09 TOUR GUIDE** is skillfully written in a friendly and easy-to-read style. Just perfect for those new to computers and to BASIC09. It's also a *valuable reference book* for programmers, engineers, students and hobbyists, providing an in-depth look at BASIC09 plus an overview of the OS-9 operating system. Comprehensive reference sections on BASIC09 and OS-9 commands are also included.

The book "maps" out your route through the Mercedes of Basics . . . BASIC09 and puts you in the driver's seat in no time. Fasten your seatbelt, sit back and enjoy the ride to perfecting your programming skills.

MICROWARE . . .

The **OFFICIAL BASIC09 TOUR GUIDE** comes from the people who wrote BASIC09. As the leader in 6809 system software, we at MICROWARE care about our users and want to help you get the most from our products.

It's Easy to Order.

Phone orders are accepted from MasterCard or VISA cardholders or for COD shipment. You can also order by mail using the coupon below. Quantity discounts are available to educational organizations and dealers. For further information contact Microware.

Microware Systems Corporation
1866 N.W. 114th Street
Des Moines, Iowa 50322
Telephone 515/224-1929
Telex 910-520-2535

Please send __________ copies of the **Basic09 Tour Guide** book at $18.95 each. Add $2.00 for UPS shipping in the U.S. or $5.00 for overseas air mail per book. Iowa residents add 4% sales tax.

Name ____________________

Address ____________________

City ____________________

State __________ Zip __________

□ I have enclosed a check

□ Charge to my bank card:
MasterCard □ VISA □
Card Number ____________________
Expiration ____________________

microware®

Specialists in system software for 68-family microprocessors since 1977.

OS-9 and BASIC09 are trademarks of Microware and Motorola.

Give Your OS-9 System The Power It Deserves!

Total Management Planning Software presents

The TMP™ POWER SERIES

The TMP FREEFORM/FILER

The Premier Text/Filing Program!

A unique text/filer program that bridges the gap between traditional data base management and word processing programs!

UNIQUE CAPABILITIES:

✓ **THE OBJECTIVE:** Designed for busy and creative individuals who want the **FREEDOM** to randomly store information, but need the **POWER** to retrieve that information in seconds.

✓ **THE KEYWORD SYSTEM:** As you enter or edit your text, you can select any word as a "KEYWORD" or "KEYPHRASE." Each "CARD" can contain as many as *117 KEYWORDS, and up to nine pages* of text similar to a 3" x 5" card, with no field restrictions. Each "FILE DRAWER" can contain up to 32,767 pages!

✓ **THE EDIT:** Includes an onscreen editing menu, word wrap, moving lines within a page, and card duplication.

✓ **THE SEARCH:** Search for the Card Title, KEYWORD, or a *combination* of both. Plus, *"WILD CARD SEARCHES"* save time!

✓ **THE RESULTS:** List the titles of cards found in the search, print the cards, or write them to a disk file for later printing or use with a word processor.

GIVE YOU MORE ABILITIES:

★ **BUSINESS:** Appointments, office phone and address indexes, inventory, service calls, vendor lists, and sales orders.

★ **ATTORNEYS:** Find the cases affected by new court decisions.

★ **DOCTORS:** List all patients affected by new medications.

★ **WRITERS AND RESEARCHERS:** For indexing, cross-referencing or cataloging.

★ **PERSONAL:** The possibilities are endless . . . art and coin collections, bill paying, income tax records, and home inventory.

NOTE: The ***FREEFORM/FILER*** integrates data into the ***"POWER MANAGER."***

'68' MICRO JOURNAL Said:

"TMP FREEFORM FILER is a flexible Program that can be used for a multitude of tasks without requiring a Computer Science background . . . The Manual is well laid out and easy to read and use . . . All in all, ***TMP FREEFORM FILER*** is a very useful Program."

Requires 64K. (Regular Retail Price: $225.)

Special Introductory Price: $195.

NEW RELEASE:

The ***TMP QFIND*** program is now included with the ***TMP FREEFORM FILER*** at no additional charge! ***TMP QFIND*** is the quick and easy way to call up information entered in card drawers and is great for addresses, phone numbers, "help" systems, contact lists, etc.!

The TMP FRONT-END

. . . Gives Your OS-9 System An Added Edge in Ease of Use!

★ ***The TMP FRONT-END*** allows the user to call TMP programs, application programs, editors, shell scripts, OS-9 commands, and other programs from a menu system.

★ The ***FRONT-END*** is a ***POWERFUL MENU PROGRAM*** that allows up to seven menu screens, each having up to 36 menu options!

★ The ***FRONT-END*** can be entered at any level and can be called from non-TMP programs. It allows the user to run the OS-9 shell directly without exiting the menu system.

★ The ***FRONT-END*** comes with sample menu scripts and instructions that make operation a snap even for first-time OS-9 users.

Requires 64K, $50.

SPECIAL PACKAGE OFFER:

Each of these programs if purchased separately would cost $890. In this special introductory offer, all four programs can be purchased for $690, a savings of more than 20 percent!

The TMP POWER PLANNER

. . . is the Leader of a New Generation In Spreadsheets!
. . . is Unequalled in Speed and Power!

POWER PLANNER Features:

★ The unique ***"POWER UPDATE"*** *which uses "circular referencing,"* a new concept that makes ***POWER PLANNER*** far superior to Other Spreadsheets.

★ The ***POWER PLANNER*** also includes the traditional row and column ordered calculations found in Other Spreadsheets. But circular referencing is much faster because it recalculates only related cells!

★ ***DATA ENTRY FREEDOM: POWER PLANNER*** eliminates worrying over the order of calculation! Data and formulas can be entered in *any order*, and you get *correct results* no matter where your information is placed!

★ With ***POWER UPDATE*** you always get the correct answer, unlike Other Spreadsheets that force you to keep recalculating until you get the right answer.

★ ***SPEED:*** For handling large spreadsheets, the ***POWER PLANNER*** is the fastest. We invite comparisons!

★ ***"POWER OVERLAY"*** lets you easily create ***Specialized Reports*** with "Boiler Plate" screens. This means you can ***overlay any number of screens*** from disks containing formulas, constants, or formats, and automatically update one spreadsheet with another!

★ ***EASE OF USE:*** Fewer Keystrokes, Help screens, and "Snap Shot" printing.

★ A worksheet that will display up to ***254 rows by 255 columns.***

★ ***Full 13-digit precision,*** includes all the standard arithmetic and trig functions, such as sum, average, maximum, sine, pi, etc.

For Speed and Power for Management Decisions in Budgeting, Cash Flow Comparisons, Expense Statements, Sales Forecasts, Profit/Loss Projections, Tax, Loan and Financial Analysis, and all kinds of "What If" Calculations, ***there is no other spreadsheet that is equal to the POWER PLANNER!***

Requires 64K, $250.00

The TMP POWER MANAGER

. . . Best In Its Class!

POWERFUL CAPABILITIES:

✓ More ***characters per record (7500)*** than any other program in its class!

✓ Each database can contain ***32,767 records.***

✓ Each record can contain up to ***150 fields of data.***

✓ Each field can contain up to ***50 characters.***

✓ Fields can be ***alpha, numeric, date, or monetary.***

✓ Powerful ***Sort and Report Generation capabilities.***

✓ ***SORTING*** on any field or on a combination of fields. Plus, extra power from the ***"OR SORT"*** and descending sort features.

✓ Intricate math ***CALCULATING*** between fields based on formulas you devise.

✓ The ***POWER MANAGER*** comes with an extensive Tutor.

GIVE YOU MORE ABILITIES:

★ ***POWER MANAGER*** can create ***CUSTOMIZED LETTERS, INVOICES,*** columnar reports, or label formats for mailing!

★ Use the ***Report Generator*** feature in conjunction with a text editor or word processor to write personal letters to those individuals who meet the ***geographic, date or monetary criteria you select!***

★ Our Users put ***POWER MANAGER*** to work for them to do: . . . customer mailings . . . past due notices . . . invoicing . . . sales analysis . . . order tracking . . . inventories . . . credit, insurance and employee records, client profile reports, and much more.

► ***THE POWERFUL BOTTOM LINE:*** We take the competition head on, and dollar for dollar . . . The ***POWER MANAGER*** is by far the best in its class!

Requires 128K, $365.00

► ORDERING INFORMATION: TMP SOFTWARE • 2431 E. Douglas • Wichita, Kansas • 67211

► OR CALL TOLL-FREE: 1-800-255-1382 Ext.47

We accept VISA, MC, AMEX, money orders and checks.

NOTE: The parent company of TMP Software, The United Software Co., is now the distributor and support organization for TMP Software.

More help than any other thing I have ever read, except 68 Micro Journal, of course. Come on with the C book on Norm Commo's column.
BD

FLEX™ USER NOTES

THE 6800-6809 BOOK

By: Ronald W. Anderson
As published in 68 MICRO JOURNAL™

SWELL! - Best buy EVER in a 6800/6809 hook.
JTW

The publishers of 68 MICRO JOURNAL are proud to announce the publication of Ron Anderson's **FLEX USER NOTES,** in book form. This popular monthly column has been a regular feature in 68 MICRO JOURNAL SINCE 1979. It has earned the respect of thousands of 68 MICRO JOURNAL readers over the years. In fact, Ron's column has been described as the 'Bible' for 68XX users, by some of the world's leading microprocessor professionals. Now all his columns are being published, in whole, as the most needed and popular 68XX book available. Over the years Ron's column has been one of the most popular in 68 MICRO JOURNAL. And of course 68 MICRO JOURNAL is the most popular 68XX magazine published.

As a SPECIAL BONUS all the source listing in the book will be available on disk for the low price of: FLEX™ format only — 5" $12.95 — 8" $16.95 plus $2.50 shipping and handling, if ordered with the book. If ordered separately the price of the disks will be: 5" $17.95 — 8" $19.95 plus $2.50 shipping and handling.

Listed below are a few of the **TEXT** files included in the book and on diskette.

All TEXT files in the book are on the disks.

LOGO.C1	File load program to offset memory — ASM PIC
MEMOVE.C1	Memory move program — ASM PIC
DUMP.C1	Printer dump program — uses LOGO — ASM PIC
SUBTEST.C1	Simulation of 6800 code to 6809, show differences — ASM
TERMEM.C2	Modem input to disk (or other port input to disk) — ASM
M.C2	Output a file to modem (or another port) — ASM
PRINT.C3	Parallel (enhanced) printer driver — ASM
MODEM.C2	TTL output to CRT and modem (or other port) — ASM
SCIPKG.C1	Scientific math routines — PASCAL
U.C4	Mini-monitor, disk resident, many useful functions — ASM
PRINT.C4	Parallel printer driver, without PFLAG — ASM
SET.C5	Set printer modes — ASM
SETBAS1.C5	Set printer modes — A-BASIC
	(And many more)

**Over 30 TEXT files included in ASM (assembler) — PASCAL — PIC (position independent code) TSC BASIC-C, etc.

NOTE: .C1,.C2, etc.=Chapter 1, Chapter 2, etc.

This will be a limited run and we cannot guarantee that supplies will last long. Order now for early delivery.

Finally I don't have to type in all that source code, a GREAT, GREAT idea. Thanks and good luck, you guys are the best!
TOA

Foreign Orders Add $4.50 S/H

Softcover — Large Format

Book only: **$7.95** + $2.50 S/H

With disk: 5" **$20.90** + $2.50 S/H

With disk: 8" **$22.90** + $2.50 S/H

I will keep this (FLEX USER NOTES) book always beside my computer. I have not ever, with any magazine, received as much information as this one book, and 68 MJ.
DIB

See your local S50 dealer/bookstore or order direct from:

**Computer Publishing Inc.
5900 Cassandra Smith Rd.
Hixson, TN 37343
(615) 842-4601**

TELEX 558 414 PVT BTH

™FLEX is a trademark of Technical Systems Consultants

Flex User Notes

Ronald w. Anderson
3540 Sturbridge Court
Ann Arbor, MI 48105

Once More

I have from time to time in this column made a plea that I wish you out there would heed. I don't mean to be harpy or negative in saying this, but it is time for a repeat. This column is my donation to the 68XX users. It is done without compensation other than to be able to keep the software that I review. I enjoy it greatly, though at one time I had considered stopping in order to devote more time to writing freelance for pay. On long consideration, I decided that if I were to get paid for a regular column, it would cease to be any fun, and would turn into hard work. On that basis, I continue (haven't missed a month in 5 years now).

Since this is a donated effort, you must realize that I work a full time job in addition to this. The column does not really take all that much of my time to write (as some of you realize by my sometimes very disorganized writing). Now to the point. I would REALLY appreciate it if readers would not take advantage of the fact that my phone number is listed by calling me for help, advice, or answers to questions. As I have said before, your call forces me to give you my attention at YOUR convenience. If you want advice or help with something and you think I might be able to give you a hand, PLEASE take the time to write a letter. I need to add a plea to see that your address is somewhere on the LETTER, not just on the envelope. Envelopes tend to get lost or thrown away, and a couple of times when I was getting at answering a letter, I have found that I had no address to which to send the reply. (I hasten to add here that I don't have all the answers, and that some letters do, now and then, fall in a crack somewhere and do not get answered.) The advantage to you is that I can spend more time thinking about my reply. Perhaps I will see something of interest for all the readers and spend some time writing some code for you, that we can all share. The advantage to me is that I can answer your letter at MY convenience some night at midnight or I AM, when most of you wouldn't dare call me.

If you do call, be prepared for a very cool answer. A few weeks ago, I had a particularly trying week at work, and I had spent about 5 hours on the phone with various computer related conversations, all in the evening. When one poor unsuspecting reader called me on Saturday as I was on my way out the door on an errand, I was abrupt and just plain rude. The reader, of course had no way of knowing the circumstances, but he was kind enough to send me a letter outlining what he wanted to discuss on the phone. His response was not unkind in any way, and I am going to discuss his thoughts here. I'm sorry Kent for the way I answered the phone that Saturday. I apologize publicly.

COCO Again

The letter was from Kent Meyers of Le Roy MN. It was in response to my discussion of the COCO in the November '84 Micro Journal. I had agreed with a reader's letter about Radio Shack providing poor documentation. Kent reminded me of the very nice service manuals available from Radio Shack for the computer and the old disk interface and pointed out that perhaps the reader who wrote to complain about lack of documentation on the new improved disk interface had bought the interface too soon and that the documentation simply was not yet published. Kent, you could be correct on that, and in fact probably are. I appreciate being reminded that the hardware documentation is available.

I could say something here about the software documentation being disorganized. Try finding out how to use the matching language routines mentioned in the manual in connection with reading and writing to a tape cassette, for instance. It seemed to me at the time I was looking at that, that Radio Shack included or left out information more or less by chance rather than by careful consideration.

Kent goes on to say that I was "off the mark on the question of using an 80 column terminal on the COCO. The problem of missed characters only shows up in the screen editors....". Kent indicates that he uses a line editor with the COCO. My goodness! I thought people quit using line editors 5 years ago. I certainly haven't used one to any extent for at least that time. Kent is saying that I have to use an external terminal on the CoCo to get more than 51 characters on a line, and then I have to use a line editor? Forget it. I'm too spoiled by my SS-50 system! (Kent, there is the problem with the CoCo and I. If you don't have a more capable system the CoCo is great for learning about computing and computers.)

Kent says that the Tano Dragon corrected all the faults of the CoCo. It used a built in 6551 ACIA "that completely cures the missed character syndrome". "Those who got the Dragon at the close-out price of $139 were very lucky." Great! We have a computer that solves the CoCo's problems but the manufacturer is not supplying it anymore.

"In the first paragraph about OS9, you use "multi-tasking" in reference to the use of an external terminal. This is more properly termed "multi-user". When I first got my copy of OS9, the first thing that I attempted was using the external terminal. When I discovered that it would not run reliably above 300 baud, I put OS9 on the shelf for a while... Now that I have the Dragon, though, I have begun to get into OS9... at 9600 baud with no missed characters."

I get the feeling that Kent is agreeing with my conclusions but not my terminology here. If I put an external terminal on the CoCo so I can be the single user, how can you call that "multi-user"? I know what you are saying Kent, but the fact that OS9 won't let me switch to a single user mode is part of the problem. OS9 insists on running its CLOCK task and I can't shut off the "console" task that keeps the COCO keyboard live. I did find that I could reduce its allotted time slice to minimum for that task and run pretty reliably at 600 baud with an external terminal. The fact that the Dragon works so well is great for anyone who owns one, but sort of irrelevant to you COCO owners who don't have a Dragon.

You may recall that I did conclude the discussion by saying that the COCO is NOT a useless toy as some say it is. It is an excellent and inexpensive way to learn something about computing and computers. I said that you could try computing on for a fit, and if you didn't like it you wouldn't be out a large wad of money. Kent points out an additional nice feature of owning a COCO. With reference to that, Kent said "If he shops wisely, when he gets rid of his COCO all of his peripherals (disk drives, printer, etc.) can be transported and used on his new system. The only Color Computer specific components would be the computer itself and the disk controller."

Kent concluded his letter by saying that he wouldn't bother me by phone again. I replied "now if I could only get several hundred other people to understand...."

Used Equipment

I recently found a used Centronics 737 printer for a reasonable price. It has a proportionally spaced type font and it is one of the printers for which STYLO is configurable. It does an excellent job with text via STYLO. Only problem with the 737 is that for some unknown reason, Centronics didn't bother to build it so it would recognize a formfeed character and feed to the top of the next page. Good Grief, a printer that does beautiful proportional spacing of text and won't page! Printer Driver to the rescue! I spent the evening writing a printer driver that counts linefeeds and feeds to the top of the next page when a $0C (formfeed) command is sent to it. That is, if the driver is on line 50 and it sees a formfeed, it outputs 16 linefeed characters to feed the paper to the top of the page. The printer driver initializes with a line count of zero. If you power up with the paper properly aligned, you never could guess that the printer doesn't honor formfeeds. While I was at it, I threw in the left margin feature that I had done in a printer driver before, and wrote a PAGE command for the printer so you can always feed a page to tear off a listing that is in the printer. On the basis that two or three of you might want this utility, I'll send hard copy to anyone who writes and asks, or copy the source to a disk you send me, with return postage. I have a version of FLEX that has a hole at $CB00 so I put the driver overflow there. I still use the "old" two file method for printing. The P.CMD file loads the PRINT.SYS file. I know, some of you laugh at this, but it nicely separates the things that are common to all print drivers out into the P.CMD so that my "individual" print drivers can be short and simple. I see no reason to change the system but you can add the functions of P.CMD to my PRINT.SYS file if you like.

I now have print drivers for IDS Paper Tiger, Heath H-14(serial), Epson MX-80 (or 100), RX-80 (not the same as MX-80), Centronics 737 and a couple other printers that have come and gone here and at work.

Still in the line of used equipment, I got a friend and a used SS-50 bus computer together recently. The friend didn't want to spend the price of a terminal in addition to the computer, so he found a used GIMIX Video board. I volunteered to get it working with a monitor and a serial output keyboard that was available.

It soon became apparent that I would have to interface the video board at the monitor level (i.e. modify SBUG-E to jump to the appropriate routines for outputting a character to the terminal, inputting a character with echo, and for initializing the board on power up. I succeeded reasonably well in these goals. First, the board needs 2K of memory somewhere. I had added enough decoding to the old mother board so I could put the video RAM at $E800 to $EFFF. The control ports conveniently fit on the four address per port system at $E020. They would fit on a 16 per port system at $E200. I decided to experiment with drivers written in PL9 and first simply wrote a test program to input characters from the terminal and echo them to the video board. The board is not the ultimate in convenience like some of the newer boards that look just like a terminal. It has a cursor row register, a cursor column register, a "last row" register and a general control register. You must program the character set to be used (though one is in ROM so you don't have to design a type font).

I found it straightforward to write a fairly large PUTCHAR procedure that handles the common case of outputting a printable character in an express mode and takes more time as required to look at control characters and handle them properly. The cursor position bears no relation to where on the screen the next character will appear. That is only a function of which memory location you write into. Keeping the two in sync at end of lines, when wrapping around from end to start, and controlling scrolling via the last line register were a little challenge, though the greatest challenge was to make the routine character handling as fast as possible. When I was done, the test was to fill the screen with characters while in a screen editor and then move the window by one screen. The screen was rewritten in just about 1 second, so I figure the effective character handling to be near 19,200 baud, up there with the fastest of serial terminals. The drivers have software to interpret commands to place the cursor anywhere on the screen, clear the screen, backspace, and erase from cursor to the end of the current line. I made the commands the same as one of my terminals so I can help the friend configure his software to work properly with it.

STAR-DOS

I understand that a review of STAR-DOS is underway, which may or may not appear before this column, so this is NOT a review. However, I must give you my impressions of STAR-DOS. About a month ago a package arrived in the mail containing a disk on which was a version of STAR-DOS that would run on one of my computers. Peter Stark had sent it to me for comments and/or the finding of any remaining incompatibilities with FLEX. For those of you who have not been following the ads, I had better back up a little. STAR-DOS is Peter Stark's answer to FLEX. It was written from scratch, but done in such a way that nearly all software that was written to run under FLEX will run under STAR-DOS. In case you hadn't guessed, STAR KITS and Peter Stark are synonymous.

I unpacked the disk and booted STAR-DOS to find a strange prompt in place of the familiar +++ that I have been looking at for so long. STAR-DOS: was the prompt that I saw. Any of you who have followed my column and my reviews for any length of time know my "luck" at finding bugs in software the first time I try to run it. STAR-DOS was no exception in the broad sense... What I mean is that there were still some differences between what STAR-DOS does and what FLEX does. For example, the first thing that struck me was that I couldn't specify a filename as I have always done with FLEX as FILENAME.EXT.1. STAR-DOS insisted on the other form acceptable to FLEX, namely 1.FILENAME.EXT. Peter happened to call me and ask if I had gotten the package and when I mentioned this to him he said something like "I've always done it the other way. I didn't know FLEX accepted the drive number last." When I assured him that the flex manuals have documented both forms of file specifications ever since the first Miniflex manual I have, he indicated that he would fix STAR-DOS to accept either form, and he has done that.

I had a couple other difficulties not worth mentioning since they too have been fixed. When I complained about the STAR-DOS: prompt and indicated that I wished it possible to change it to *** Peter wrote a "PROMPT" utility that lets you change it to whatever you like. I've been impressed at Peter's willingness to track down and "adjust" any little difference between STAR-DOS and FLEX with regard to how they handled various situations. I use the word "adjust" because "fix" might imply a problem, and the little things I found were not problems, just differences between the way the two operating systems handle certain situations.

STAR-DOS comes complete with an assortment of disk utilities and a user manual. There is an "installation" manual available at extra cost for those users who want to install STAR-DOS on different 6809 hardware, i.e. with different I/O port addresses or driver requirements. That manual is a very well written step by step procedure for getting STAR-DOS up and running on any 6809 hardware, and with any disk controller. Of course, a project as involved as writing new disk drivers is not for the computer beginner, but someone with some experience programming in Assembler would have no trouble getting the job done.

Peter has implemented one cute feature that he (and I) have found to be valuable. If you have a calendar clock board in your computer and can write an assembler program to read hours and minutes and convert them to one byte that codes hours and tenths on a 24 hour basis (eg. 10:30 PM would be 225, 22 for the hour and 5 for the .5 hour) you can link that code to STAR-DOS so that whenever you write a file to the disk it will have not only the normal date information, but the time as well (to within 6 minutes). A utility that takes advantage of this is supplied with STAR-DOS. It is called TCAT. TCAT is a catalog utility that lists the files on the disk in reverse order of date and time. The result is that the last file you worked on is listed at the top of the catalog! If you are like me, and hit the sack at 2:30 AM after working on computing projects, the next night you say "let's see, what was I working on last?" You type TCAT and there it is before your eyes. Nice...

Though at this point I am quite sure there are not very many incompatibilities left, there may still be a few. What is impressive to the point of outweighing the possibility of a few incompatibilities, is Peter's eagerness to "adjust" these. There is one "problem" that remains that could be a deterrent to your being interested in STAR-DOS just yet. It does not support random files at this point. The next version will have that feature added. If you run software that uses random files, you might want to hold off on STAR-DOS for a while.

I should mention that Peter has several versions of STAR-DOS designed to run with various disk controller configurations. One in particular is very nicely compatible with the Peripheral Technology PT-69. When I mentioned that I had a PT-69 running, Peter sent me that version to try out, and it works quite nicely.

Now to get into trouble with TSC... Though the good folks at Technical Systems Consultants who wrote FLEX a few years back have staunchly maintained that they intend to "support" FLEX for the foreseeable future, it seems to me that they have done little more than elevate the price for FLEX to new heights. I can think of only one new software product from them that runs under FLEX that has been released in the past two years. That is their Relocatable Macro Assembler. It was released in Uniflex version a couple of years before the FLEX version finally became available. Let's face it. TSC has gone on to bigger and better things, and more power to them in their new endeavors. May they all get rich and famous. Had it not been for FLEX, the 6809 would be known only to industrial users running Motorola development systems writing programs for control applications.

The facts of the present are, however, that TSC has all but abandoned the FLEX market. Virtually all of the new software products to run under FLEX are from new and/or small software companies. To be a little realistic about things, as long as they can sell a copy of FLEX for $250 without even advertising it, they are not going to announce that they are no longer supporting it. Now we have an alternative operating system and at a very attractive price, including very fine documentation.

See STAR-KITS or S.E. Media advertising, for pricing and ordering information - **STAR-DOS/STAR-DOS+.**

BBS Systems

A further note from Kent Meyers included the following. "I would also like to see some mention made in the Micro Journal of the only two BBS systems in the United States that are dedicated to the dissemination of Public Domain Software for 68XX computer systems. One is located in Oklahoma City, OK and the other is in Hawthorne, CA. The Flexnet system in Oklahoma City has been on line for about two years (405-728-7654) and has a special file transfer protocol to allow error free transmission of files. The California system is just coming on line (213-539-7619) and plans to offer both the Flexnet and the CP/M Xmodem protocols for file transfer. The interesting thing about this system is that it is running on a CP/M computer."

RX and TX

I've done a few more things to the programs USEND and UREC published here a couple of columns ago. In using them for a while, it became apparent that they were a little inconvenient in several ways. First of all, if I had several files to transmit, I had to go back and forth between the two system's terminals to type in all the proper commands. First step was to add some code to interpret a list of files on the command line at the transmit end and to transmit the filename to the receive end. I simply set up the File Control Block at the transmit end and sent the 11 characters starting at the 5th (filename and extension) to the receive end where they were placed in an FCB. The receive end then attempts to open the file and reports to the CRT the fact that the file already exists, asking if it may be deleted. That was fine when I didn't attempt to send a string of files to update existing ones on the receive end, which still necessitated typing on both terminals. The final improvement was to send the delete question back to the sending end and allow it to be answered from there. I also found that my input routines neglected to mask off the parity bit, which went undetected until I happened to try them on a system with a terminal that had parity enabled. The character input GETCHAR and GETIF routines are now properly set up to AND off the parity bit just in case.

Now I can set up a string of filenames in the command line and go at copying them from system to system. I only have to type RX at the receive end, and RX stays alive until it sees an escape at the receive end. I can therefore use TX a number of times without ever going near the receive end.

I'll include the present listings of these utilities here for your information or use. I am not yet quite finished with them, however. The final version will read the directory at the send end and do string matching just as the various versions of copy utilities do. In other words, TX .TXT will transmit all files with the extension .TXT. TX TEST will transmit TEST1, TEST2, TESTFILE, etc. I didn't mention above that presently the filenames appear on the terminals at each location as the files are opened.

I suppose that out of the SS-50 bus owners that read this, perhaps a couple hundred of you will have PL/9 and of those, two or three might have two systems. If anyone does actually use these, I'd like to know what you think of them. I think they are a good demonstration of programs that access the hardware of the system without resorting to assembler code in any way.

- - -

```
   *
/* PROGRAM TO SEND A TEXT OR BINARY FILE VIA WIRE.
   BINARY FILE IS CONVERTED TO ASCII FOR TRANSMISSION.
   THIS HAS BEEN TESTED UP TO 9600 BAUD TRANSMISSION RATE.
*/

ORIGIN = 0;
STACK = $BDFF;

CONSTANT
   CR  = $0D,
   LF  = $0A,
   BIN = 'B,      /* THESE ARE USED FOR CONTROL HANDSHAKING */
   START = 'S,  /* THE ONES THAT ARE NOT USED EMBEDDED IN THE */
   TXT = 'T,      /* FILE STREAM HAVE BEEN MADE PRINTABLE CHARACTERS */
   DELRQ = 'D,   /* FOR EASE OF DEBUGGING. */
   SENDRQ = '+,
   FILERQ = 'F,
   READY  = 'R,
   END = $06,    /* LAST TWO MUST BE CONTROL CHARACTERS */
   WAITING = $05;

AT $E000 BYTE MODEM(2);
AT $E004 BYTE TERM(2);
```

```
INCLUDE TRUFALSE.DEF.1;
INCLUDE FLEX.LIB.1;


PROCEDURE PUTCHAR(BYTE .DEVICE: BYTE CHAR);
   REPEAT UNTIL DEVICE(0) AND 2
   DEVICE(1)=CHAR;
ENDPROC;


PROCEDURE CRLF;
   PUTCHAR(.TERM,CR);
   PUTCHAR(.TERM,LF);
ENDPROC;


PROCEDURE PRINT(BYTE.DEVICE, .STRING): BYTE N;
   N=0;
   WHILE STRING(N)
   BEGIN
      PUTCHAR(.DEVICE,STRING(N));
      N=N+1;
   END;
ENDPROC;



PROCEDURE GETCHAR (BYTE .DEVICE);
   REPEAT UNTIL DEVICE(0) AND 1;
ENDPROC BYTE (DEVICE(1) AND $7F);


/* GETIF RETURNS CHAR IF ONE HAS BEEN INPUT, ELSE RETURNS NULL.
   THIS PROCEDURE DOES NOT WAIT FOR A CHARACTER  */


PROCEDURE GETIF (BYTE .DEVICE);
   IF DEVICE(0) AND 1 THEN RETURN BYTE (DEVICE(1) AND $7F);
ENDPROC BYTE 0;


PROCEDURE BINASC(BYTE CHAR, .UNIBBLE, .LNIBBLE);
   LNIBBLE = CHAR AND $0F;
   UNIBBLE = SHIFT(CHAR,-4) AND $0F;
   IF LNIBBLE < 10 THEN LNIBBLE = LNIBBLE + $30
      ELSE LNIBBLE = LNIBBLE + $37;
   IF UNIBBLE < 10 THEN UNIBBLE = UNIBBLE + $30
      ELSE UNIBBLE = UNIBBLE + $37;
ENDPROC;


PROCEDURE SEND_FILENAME(BYTE .FCB):BYTE N,CHR;
   N=4;
   WHILE N< 15
   BEGIN
     CHR = FCB(N);
     PUTCHAR(.MODEM,CHR);
     IF CHR=0 THEN PUTCHAR(.TERM,$20) ELSE PUTCHAR(.TERM,CHR);
     IF N=11 THEN PUTCHAR(.TERM,'.');
     N=N+1;
   END;
   PUTCHAR(.TERM,$20);
ENDPROC;



PROCEDURE SEND ( BYTE CH, CH1, CH2, BIN_FILE, INFILE(320), N:
                        INTEGER COUNT;

/* INITIALIZE MODEM PORT */
   MODEM(0)=3;
   MODEM(0)=$15; /* INITIALIZE DIVIDE BY 16 CLOCK */
   CRLF;

   REPEAT
      REPEAT
         GET_FILENAME(.INFILE);
         IF CCR AND 1 THEN        /* NO MORE FILES ON INPUT LINE. */
         BEGIN
            CRLF;
            PRINT (.TERM,"FINISHED TRANSFER");
   FLEX;
END;
SET_EXTENSION(.INFILE,1); /* DEFAULT TEXT FILE */
PUTCHAR(.MODEM,SENDRQ); /* REQUEST TO SEND */
REPEAT
CH = GETCHAR(.MODEM);
UNTIL CH =FILERQ;  /* LOCK UP HERE IF RECEIVE NOT READY */
SEND_FILENAME(.INFILE);
REPEAT CH = GETIF(.MODEM) UNTIL CH <> 0;
IF CH = DELRQ THEN
BEGIN

      PRINT (.TERM,"DELETE EXISTING FILE? ");
      CH = GETCHAR(.TERM);
      PUTCHAR(.TERM,CH);
      PUTCHAR(.TERM,$20);
      PUTCHAR(.MODEM,CH);
      IF CH <> 'Y .AND CH <> 'y THEN
      BEGIN
         CRLF;
         CLOSE_FILE(.INFILE);
      END;
   END;
UNTIL CH = READY .OR CH = 'Y .OR CH = 'y;
IF CH <> READY THEN REPEAT CH = GETIF(.MODEM) UNTIL CH = READY;
OPEN_FOR_READ (.INFILE);
IF INFILE(1)<>0 THEN
BEGIN
   REPORT_ERROR(.INFILE);
   FLEX;
END;

CH = READ(.INFILE);
IF CH = $02 THEN
BEGIN
   SET_BINARY(.INFILE);
    BIN_FILE = TRUE;
   PUTCHAR(.MODEM,BIN);
END
ELSE
BEGIN
   BIN_FILE = FALSE;
   PUTCHAR(.MODEM,TXT);
END;
REPEAT CH2= GETCHAR(.MODEM) UNTIL CH2= START;
IF BIN_FILE THEN BINASC(CH, .CH1, .CH2);
COUNT = 0;

WHILE INFILE(1)=0
BEGIN
   REPEAT
      IF BIN_FILE THEN
      BEGIN
         PUTCHAR(.MODEM,CH1);
         PUTCHAR(.MODEM,CH2);
         COUNT = COUNT+2;
      END
      ELSE
      BEGIN
         PUTCHAR(.MODEM,CH1;
         COUNT = COUNT + 1;
      END;
      IF COUNT AND $00FF = 0 THEN PUTCHAR(.TERM, '*);
      CH = READ(.INFILE);  .
      IF BIN_FILE
         THEN BINASC(CH, .CH1, .CH2);
   UNTIL COUNT = 10000 .OR INFILE(1)<>0
   IF INFILE(1)=0 THEN
   BEGIN
         PUTCHAR(.MODEM,WAITING);
         COUNT = 0;
         REPEAT UNTIL GETIF(.MODEM)= START;
      END;
   END;
   IF INFILE(1)<>8 THEN REPORT_ERROR(.INFILE);
   CLOSE_FILE (.INFILE);
   PUTCHAR(.MODEM,END);
   CRLF;
   REPEAT CH=GETCHAR(.MODEM) UNTIL CH = START;
 FOREVER;
```

```
/* PROGRAM TO RECEIVE A TEXT OR BINARY FILE VIA WIRE.
   BINARY FILE HAS BEEN CONVERTED TO ASCII BY SEND PROGRAM, AND
   THIS PROGRAM CONVERTS IT BACK TO BINARY FORM.
   PROGRAM HAS BEEN TESTED UP TO 9600 BAUD WITH NO DIFFICULTIES
*/

ORIGIN = 0;
STACK = $BOFF;

CONSTANT
   CR  = $0D,
   LF  = $0A,
   BIN     = 'B,
   START   = 'S,
   TXT     = 'T,
   SENDRQ  = '*,
   FILERQ  = 'F,
   READY   = 'R,
   DELRQ   = 'D,
   END     = $06,
   WAITING = $05;

GLOBAL
   BYTE OUTFILE(320);

AT $1000 BYTE DATA(10010);
AT $E000 BYTE MODEM(2); /* 6850 MODEM PORT ADDRESS */
AT $E004 BYTE TERM(2);  /* 6850 TERMINAL PORT ADDRESS */
AT $CC0C BYTE WRKDRV;   /* WORKING DRIVE NUMBER */

INCLUDE TRUFALSE.DEF.1;
INCLUDE FLEX.LIB.1;  /* FLEX FILE HANDLING INTERFACE */


PROCEDURE PUTCHAR(BYTE .DEVICE: BYTE CHAR);
   REPEAT UNTIL DEVICE(0) AND 2;
   DEVICE(1)=CHAR;
ENDPROC;


PROCEDURE GETCHAR (BYTE .DEVICE);
   REPEAT UNTIL DEVICE(0) AND 1;
ENDPROC BYTE (DEVICE(1) AND $7F);


PROCEDURE PRINT(BYTE .STRING):BYTE N;
   N=0;
   WHILE STRING(N)
   BEGIN

      PUTCHAR(.TERM,STRING(N));
      N=N+1;
   END;
ENDPROC;


PROCEDURE CALF;
   PUTCHAR(.TERM,$0D);
   PUTCHAR(.TERM,$0A);
ENDPROC;


PROCEDURE ASCBIN(BYTE UNIBBLE, LNIBBLE);
   IF UNIBBLE >$40 THEN UNIBBLE = UNIBBLE - $37
      ELSE UNIBBLE = UNIBBLE - $30;
   IF LNIBBLE >$40 THEN LNIBBLE = LNIBBLE - $37
      ELSE LNIBBLE = LNIBBLE - $30;
ENDPROC BYTE(SHIFT(UNIBBLE,4) + LNIBBLE);


PROCEDURE GETFILE_MODEM: BYTE N, CHAR;
   PUTCHAR(.MODEM,FILERQ);
   OUTFILE(3)=WRKDRV;
   N=4;
   WHILE N < 15
   BEGIN
      CHAR = GETCHAR(.MODEM);
      OUTFILE(N)=CHAR;
      IF CHAR = 0 THEN PUTCHAR(.TERM,$20) ELSE PUTCHAR(.TERM,CHAR);
      IF N = 11 THEN PUTCHAR(.TERM,'.);
```

```
      N=N+1;
   END;
   PUTCHAR(.TERM,$20);
ENDPROC;


PROCEDURE GETIF(BYTE .DEVICE);
   IF DEVICE(0) AND 1 THEN RETURN BYTE (DEVICE(1) AND $7F);
ENDPROC BYTE 0;

/* MAIN PROGRAM STARTS HERE */

PROCEDURE REC_LOCAL : BYTE CH, CH1, BIN_FILE:
                      INTEGER INDEX,LIMIT,DBYTE;
   CRLF;
   MODEM(0)=3;
   MODEM(0)=$15; /* INITIALIZE DIVIDE BY 16 CLOCK */
   REPEAT
      REPEAT
         REPEAT
           CH = GETIF(.MODEM);
           IF CH = 0 THEN
              CH = GETIF(.TERM);
         UNTIL CH = SENDRQ .OR CH = $1B;
         IF CH = $1B THEN
         BEGIN
           CALF;
           PRINT "EXIT RECEIVE MODE";
           CRLF;
           FLEX;
         END;
         GETFILE_MODEM;
         OPEN_FOR_WRITE (.OUTFILE);


   IF OUTFILE(1)=3 THEN
   BEGIN
      PRINT("DELETE FILE? ");
      PUTCHAR(.MODEM,'D); /*  DELETE FILE?*/
      CH = GETCHAR(.MODEM);
      PUTCHAR(.TERM,CH);
      CRLF;
      IF CH = 'Y .OR CH = 'y THEN
      BEGIN
         DELETE_FILE (.OUTFILE);
         OPEN_FOR_WRITE(.OUTFILE)
      END ELSE CLOSE_FILE(.OUTFILE);
   END;
UNTIL CH = SENDRQ .OR CH = 'Y .OR CH = 'y;

IF OUTFILE(1)<>0 THEN REPORT_ERROR(.OUTFILE);
PRINT "READY TO ACCEPT "; CRLF;
INDEX = 0;
LIMIT = 0;
PUTCHAR(.MODEM,READY);  /* TELL SENDER READY */
CH = GETCHAR(.MODEM);
IF CH = BIN THEN
BEGIN
   SET_BINARY(.OUTFILE);
   BIN_FILE = TRUE;
END
ELSE BIN_FILE = FALSE;
PUTCHAR(.MODEM,START);
REPEAT
   CH = GETCHAR(.MODEM); /* WAIT FOR A CHARACTER */
   CH1 = CH; /* SAVE IT */
   IF CH <> END .AND CH <> WAITING
      THEN
      BEGIN
         DATA(INDEX) = CH;
         INDEX = INDEX+1;
      END
      ELSE LIMIT = INDEX; /* ONE PAST LAST VALID CHAR */
   IF LIMIT<>0 THEN
   BEGIN
      PRINT "  WRITING TO DISK";
      PUTCHAR(.TERM,$0D);
      INDEX=0;
      WHILE INDEX < LIMIT
      BEGIN
```

```
          IF BIN_FILE THEN
          BEGIN
             CH = ASCBIN(DATA(INDEX),DATA(INDEX+1));
             INDEX = INDEX+2;
          END
          ELSE
          BEGIN
             CH = DATA(INDEX);
             INDEX = INDEX+1;
          END;
          WRITE(.OUTFILE, CH);
       END;
       PRINT"                    ";
       PUTCHAR(.TERM,$0D);
       PUTCHAR(.MODEM,START);
       LIMIT=0;
       INDEX=0;
    END;
 UNTIL CH1 = EAG;
 CLOSE_FILE(.OUTFILE);
FOREVER;
```

OS9 USER NOTES

By: Peter Dibble
517 Goler House
Rochester, NY 14620

Perspective

Depending on my frame of mind it's right before final exams or between Thanksgiving and Christmas. In the before finals frame I know there's no time to write a program for this column. The other frame tells me that this is a good time to think about where OS-9 has been, and where it's going.

Only one important product has come from Microware this year, OS-9 68K. I don't have a computer with a 68000, and it's too early for my network to have pulled in any solid information. OS-9 68K should have an important influence on the future of OS-9, so I'll comment on it in this column, but I'll be mostly guessing.

Only one non-Microware program comes to mind as an important new program, Sculptor (used to be Sage). I don't have it either, but I have played with it. If it were inexpensive, Sculptor could have an important effect on OS-9. Sculptor is, however, very expensive. **

Since Thanksgiving was in the recent past I'll count blessings.

There must be something special about OS-9 and the systems it runs on. We haven't all run off to the MS-Dos camp.

One nice thing about OS-9 is obvious from where I sit. On my left is my CoCo. It cost about $1000 and is mostly compatible with the $10,000 Gimix on my right. The CoCo moves along nicely. The Gimix is one of the most powerful microcomputers I know of. If that isn't enough (and of course it isn't), 68010 and even 68020 machines that run OS-9 68K are way beyond the twinkle in the eye stage. Just the 6809 spectrum, from the sub-megahertz CoCo to the 3-Megahertz chip that's supposed to be available next year, gives OS-9 systems an exceptional scope. The addition of the 68000 chips runs the highest performance OS-9 systems up near mainframes.

OS-9 is a lot like Unix. It borrows enough from Unix that I am sometime (momentarily) confused about which system I'm using. Still, OS-9 is not Unix or even a Unix look-alike. In a way that's sad; we can't just plug in all that Unix software. In another way it's an advantage; we have better performance than a Unix look-alike would, real-time capabilities that would be hard to duplicate under Unix, and reliability that wasn't designed into Unix. OS-9 doesn't look like Unix, but it is close enough that I can develop programs on my micro and move them to Unix with few changes. Moving programs in the other direction is almost as easy.

Usually the thin supply of software for OS-9 seems like a disadvantage, but even that problem has a bright side. Even with all the new CoCo OS-9 users counted in we're a small group. Too small for the big software houses to notice. If you need Framework and you need it NOW, sorry, OS-9 isn't for you; but if you are content with something a bit behind the cutting edge maybe this is the right place. It's a programming truism that analysis and design are the hard parts of creating a program. It's also true that the 808x class of micro-processors are tricky to program.

If you're a individual programmer you'll have a hard time beating the software houses in the PC market, but the big software houses don't come over here. The PC folks paid the bill for the analysis and design of lots of nice programs. The best of them were written in 808x assembler. If you combine the best of the MS-Dos database, or spread-sheet, or word processing, or communications programs you can avoid the analysis and trial-and-error that was done for the original programs. Then you can code up the result in assembler for the 6809 or 68000.

I don't think there are quite enough of us yet to support a few full-time free-lance programmers, but we're getting close. When we get there a few of the best OS-9 programmers should be able to make a tidy living coding up standard programs for us. For programmers this is a great opportunity. For everyone else? When the programs arrive they will be running under OS-9. OS-9 does things that MS-Dos doesn't.

The View from the Ivory Tower

I don't know much about process control. That's a shame because process control is what many OS-9 systems do. I've read about it a lot, and I know a little about the underside of the computers that control the laser fusion project at the University of Rochester's Laser Lab. This is just enough knowledge to be dangerous, but I'm going to bravely (foolishly) point at an area that OS-9 isn't in but could be.

For years only two versions of OS-9 have been available, Level One and Level Two. They are different, but the difference is mostly that Level Two can handle more processes than Level One. Now that OS-9 68K is available engineers can chose OS-9 for a whole new kind of application. Robotics and "simple" vision require serious number crunching and large amounts of memory. These were out of the range of the 6809 but the 68000 can reach well into the simpler areas.

I surely don't need to tell people about the power of the 68000. What the hardware vendors don't seem certain about is the advantages of combined 68000/6809. Multiple 6809's are well understood. I/O processors make an important difference in throughput on 6809-based systems. They'll be at least as important on 68000's.

The suggestion I'd like to make is that a 6809 in a 68000-based system need not stop at simple I/O processing. A heavy interrupt load hurts the performance of a 6809 and it is good at handling them. The 68000 isn't nearly as good with interrupts as the 6809. So interposing a 6809 between a 68000 and as many sources of interrupts as possible is a good idea.

Instead of making a list of all the areas where a 6809 is more cost effective than a 68000 let me suggest a job they could do nicely as a team: I don't like sorting my returnable bottles and cans out by what used to be in them. A machine should be able to do it for me. The machine would watch a conveyor belt running by it. It would find each container as it went by and figure out what used to be in it, then it would knock it into the appropriate bin.

There are two parts to this problem: vision and manipulation. The vision part is hard. The manipulation (getting the container off the belt and into the bit) relatively easy. Just finding bottles and cans lying on a belt at any odd angle would be challenging for a 68000 if the job had to be done quickly. Recognizing characteristics like shape and features on the label might put the problem right out of a 68000's range. The conveyor belt will probably have to be slowed until the program can recognize most of the containers as they go by. Operating a tool that knocks containers into bins would be easy for a 68000 fast enough to handle the vision problems, but that processor is already running flat out peering at the passing rubbish. A 6809 could easily snag information from the 68000 and babysit the stepper motors and solenoids.

Here's the trick. There's nothing in the OS-9 model of the world that says that all processes have to be running on the same processor or using only common memory. They don't even have to be running on the same kind of processor. The requirements are that some memory must be shareable between processes (for shared modules), there must be a way to copy data from one processor's memory to another's, and each process must seem to be able to reach OS-9 with a software interrupt.

I'm sure there are problems hidden in this simple idea, but think how easy it would be to work with the system. Start a 6809-object-code camera handler, a 68000-object-code vision program, and a 6809-object-code manipulator operator. Let the camera handler and the vision share a data module with the bit-image of the scene in it. Run a pipe from the vision program to the manipulator operator.

I don't know of any microcomputer system that lets you run two processors together that smoothly. There are some hardware problems, but nothing the S100 vendors haven't already dealt with. Software is the core of the problem and we are already using an operating system who's design doesn't rule out teamwork.

A Simple C Function

I hadn't planned to put any code in this month's column, but I was thinking about the trouble I had raising a process's priority from inside. Some of you might be able to use my solution. I'm afraid that this is one place where Level Two is easier to work with that Level One. I only worked it out for Level Two and the conversion will be tricky.

The problem is that OS-9 has a SVC that sets a process's priority to any number you choose, but you can't ask it to raise the priority by 10. Nor is there a simple way to ask for a process's current priority. A Level Two user has to use the F$GPrDsc SVC to get a copy of the process's entire process descriptor and pick the priority out of there. Then he can use F$SPrior to set the priority to that numbur plus the required change. That's just what one of the functions I included with this column does. The other function, Getpr, returns the current priority without changing it.

A Call from the Oracle

I just heard from Ken Kaplan. I persuaded him to make some prognostications about OS-9's future. You should view them in whatever you think is the correct light for remarks about the future of OS-9 from the president of Microware.

Ken believes that within the next two or three years OS-9 will grow to be bigger than Unix. When I sounded doubtful he admitted that his best evidence is covered by non-disclosure agreements with Microware's customers. He did point out that OS-9 is already bigger than Unix in Japan (which may become a key source of computers). If you count CoCos, Ken observed that more computers run OS-9 than Unix already (then he admitted that that's not all that significant). The important arguments stayed hidden.

Ken thinks we'll see at least two new major personal computers in United States running OS-9. Major means real mass-market machines like IBM or Sperry (those vendors are my suggestions, not his).

We can expect Fortran for OS-9 next year. It will be available on the 6809 first.

Networking for OS-9 is already available from Fujitsu on the FM-11. If I lived in Japan I could buy it today. Ken expects it to become available here soon. It sounds like pretty reasonable networking. It runs over any kind of connection and allows a user on the network to use devices on any connected computer.

- - -

** Editor's Note: Because of the expense of 'SCULPTOR' or 'SAGE' (the only name we have ever seen it run under), and the 'bugs' we saw in versions demonstrated to us, caution should be used in ordering a thousand dollar plus a couple hundred dollar piece of software.

I have received telephone reports that the latest versions are running fairly bug free. Fact is a few 'glow'. However, like all other major products advertised or reported on or in 68 Micro Journal, until we 'actually see' the fixed product, I will recommend caution to prospective purchasers.

We have received 'promises' for the past two or three years that a review copy was on the way, but as yet has not arrived. If and when a working version should arrive you can be certain that I will have a report for you. The only verified reports I have so far, indicate the software has certain failures that should not be present in such an expensive package. These reports are a couple of years old, but are all that I have from users who have purchased the product and have nothing to gain from 'hyping'. Until I know better - 'CAUTION' IS THE WORD!

DMW

- - -

```
#include <os9.h>
#define PRIORITYOFFSET 10
#define TRUE 1
#define FALSE 0

static char pdesc[512];

modpri(pid, delta)
  int pid, delta;
  {
          register int priority;
          if((priority = getpr(pid)) != -1)
                  return(setpr(pid, priority+delta));
          else
                  return(-1);
  }

getpr(pid)
  int pid;
  {
          if(getpd(pid))
                  return(pdesc[PRIORITYOFFSET] & 255);
          else
                  return(-1);
  }
getpd(pid)
  int pid;
  {
          struct registers reg;

          reg.rg_a = pid;
          reg.rg_x = pdesc;

          if(_os9(F_GPRDSC, &reg) == 0)
                  return(TRUE);
          else
                  return(FALSE);
  }
```

Raymond Casneuf
Berliner Str.54
6457 Maintal 1
W. Germany
Europe
Tel. 06181-493758

Dear Mister Dibble.

Being a Micro Journal reader from the very first hour. I really like your OS9-column very much. This also was one of the reasons for adapting OS9 to my old swtpc-box. after having used FLEX for many years meanwhile. With my adaptation of OS9 to a not supported system. I ran into some problems.
Since support of OS9 at this side of the atlantic is almost very close to zero. I wonder whether you could give my some info to complete my OS9-project?

First my hardware:
I run a swtpc system. using MP-A9 cpu board with 1 Mhz. about 150 Kram. using the VDISK-package. 2 five inch drives using the DC4-controller from Southwest (MF68). 2 eight inch drives using the DMFA2-controller also from southwest.
All of this runs under FLEX-9.0 without any problem.
Recently I bougth OS9-level 1. version 1.2 from a local authorized microware dealer. I installed the package on my system. it works great. So I have both FLEX and OS9 available now.
This all works fine with the 8 inch drives. using the DMFA2 controller.
My intension now is also to use the 5 inch drives under OS9. I might proceed in two steps: first using the drives like under FLEX by running the 'USEMF' utility. so in this way I would have 4 drives in total to my system. Next step must be to bring up a bootable system. only with the 5 inch drives.

Now the problems:
Microware doesnot support southwest products anymore. In the past there was a version available for the older DC3-controller from swtp. Unfortunately for my DC4-controller, there is nothing at all from microware.
During a recent visit in the states in bought the DC3-drivers. which from my point of view must be very similar to the DC4-drivers. Although the DC3 uses the FD1771-chip and the DC4 uses the FD1779-chip as disc-controller.

As described on the disk I created a disk with the OS9BOOT. SYS. CMDS and DEFS directories. I put the DC3 drivers. boot module and device descriptors under a separate directory. called DC3. I copied the SYSDEFS into the DEFS-directory as described.
Trying to assemble I get the message ERROR-cannot open the DEFSFILE.
So apperently I am doing something wrong
So after several evenings of frustration meanwhile I gave up (!!) for now. which finally ended up into this letter to you. hoping that you could give my some infos how to proceed.....

An other request to you would be to supply me with an example how to connect an printer to OS9 softwarewise. by using the ACIA-driver. But my minifloppy problem has priority number one. so I will not bother your with too much items at the same time.

I would very much appreciate if you could take some time to give me some helpfull info. Maybe someone at your side integrated a DC4 controller already in his system(?). who knows. And I am trying to reinvent the wheel!?

I insert some irc's. hoping this will cover at least your postage. If not please do not hesitate to let me know.

Sincerely.

Ray Casneuf

Raymond Casneuf

Dear Mister Casneuf,

I remember having problems with something called DEFSFILE when I first assembled some source I bought from Microware. Instead of explicitly including each definition file in an assembler program, they build one file containing all the necessary USE statements and simply use that file. My DEFSFILE looks like:

```
     *level set 1
     level set 2
      use /H0/DEFS/OS9Defs
      ifeq level-1
      use /H0/DEFS/OS9Sysdefs.I
      else
      use /H0/DEFS/OS9Sysdefs.II
      endc
      use /H0/DEFS/OS9IODefs
      use /H0/DEFS/OS9RBFDefs
      use /H0/DEFS/OS9SCFDefs
```

Notice that I can select the definitions files for Level One or Level Two by changing one line. If I frequently assembled programs for Level One, I'd set the level in the program instead of the defslist file. Always including all those files makes my life easy; I don't have to worry about what file includes what symbols. If you have memory problems you may want to cut down yours defslist file. Another approach would be to skip it. Remove the "use defslist" statement from the program and replace it with a few "use" statements for definitions files.

It should be easy to add support for your printer. All you need is an appropriate device descriptor. Since you mentioned ACIA, I assume your printer has a serial interface. You may need to generate a new device descriptor for the printer. First check the device descriptors you already have. If you have one called something like PR it may be a serial printer. The device P is a parallel printer by convention. If you have PR, you may need to change its device address. You can do this with debug, see the System Programmer's Manual for a map of the device descriptor so you'll know what to alter.

If you don't already have a device descriptor that almost fits, you'd be best off assembling a new one.

Page 4-9 in the System Programmer's gives an example of a SCF device driver you can follow.

I believe I have heard from other people working on support for the DC4. I would imagine that printing this letter will be the best way to let them know about you.

Pete

- - -

"C" User Notes

Edgar M. (Bud) Pass, Ph.D.
1454 Latta Lane
Conyers, GA 30207

INTRODUCTION

This month's column continues the definition of O'Keefe's string-handling library started in an earlier column.

STRING-HANDLING IN C

The "strn" family of string-handling functions does not allow arbitrary contents of strings, as the processing is controlled by the terminating nulls in each string, or by the specified length. The consistent use of this family of functions would prevent many of the problems associated with null-terminated strings which have somehow lost their terminated nulls, as the length specification provides a non-data-dependent termination condition. It may also create new problems, in that the standard C libraries assume null-terminated strings, which must be assured by the programmer.

strncat(dst, src, n) copies up to "n" characters of "src" to the end of "dst". A null character is placed at the end of "dst" if there is room.

```
char *strncat(dst, src, n)
char *dst, *src;
int n;
{
    char *save = dst;
    while (*dst++);
    for (--dst; --n >= 0; )
        if (!(*dst++ == *src++)) return save;
    *dst = '\0';
    return save;
}
```

strncmp(s, t, n) compares up to "n" characters of "s" and "t". It returns a value > 0, = 0, or < 0 when "s" > "t", "s" = "t", or "s" < "t", according to the ASCII sequence of characters. It skips the equal prefixes and uses the values of the first unequal characters to determine the comparison value.

```
int strncmp(s, t, n)
char *s, *t;
int n;
{
    while (--n >= 0)
    {
        if (*s != *t++) return *s-t(-1];
        if (!*s++) break;
    }
    return 0;
}
```

strncpy(dst, src, n) copies up to "n" characters from "src" to "dst". It returns a pointer to the beginning of "dst". Null characters are placed at the end of "dst" to fill the string.

```
char *strncpy(dst, src, n)
char *dst, *src;
int n;
{
    char *save = dst;
    while (--n >=0)
    {
        if (!(*dst++ = *src++))
        {
            while (--n >= 0) *dst++ = '\0';
            break;
        }
    }
    return save;
}
```

strnend(src, len) returns a pointer to the end of the string pointed to by "src", of no more than "len" characters.

```
char *strnend(src, len)
char *src;
int len;
{
    while (--len >=0 && *src++);
    return src-1;
}
```

strnlen(src, len) returns the number of characters up to the first null in the string pointed to by "src", or "len", whichever is smaller.

```
int strnlen(src, len)
char *src;
int len;
{
    int l = 0;
    while (--len >=0 && *src++) ++l;
    return l;
}
```

strnmov(dst, src, n) copies up to "n" characters from "src" to "dst". It returns a pointer to the next character after the end of "dst".

```
char *strnmov(dst, src, n)
char *dst, *src;
int n;
{
    while (--n >=0)
    {

        if (!(*dst++ = *src++))
        {
            src = dst;
            while (--n >= 0) *src++ = '\0';
            break;
        }
    }
    return dst;
}
```

strnrev(dst, src, len) copies up to "len" characters from "src" to "dst" in reverse order. It will work with completely overlapping, but not partially overlapping, source and destination strings. On each iteration, it swaps successive characters from the next positions from the front and end of each string.

```
strnrev(dst, src, len)
char *dst, *src;
int len;
{
    char *dstz, *srcz = src, t;
    for ( ; --len >= 0 && *srcz; srcz++);
    dstz = dst + (srcz - src);
    if (len >= 0) *dstz = '\0';
    while (srcz > src)
    {
        t = *--srcz;
        *--dstz = *src++;
        *dst++ = t;
    }
}
```

strnrpt(dst, n, src, k) repeats the string "src" into "dst" "k" times, but truncates the result at "n" characters.

```
int strnrpt (dst, n, src, k)
char *dst, *src;
int n, k;
{
    char *save = dst, *p;
    for ( ; --k >= 0; --dst)
        for (p = src; ; )
        {
            if (--n < 0) return dst-save;
            if (!(*dst++ = *p++)) break;
        }
    return dst-save;
}
```

The following routines convert integers to strings and strings to integers. The radix of the integer (2-36) is specified on each conversion. In case of error in conversion from string to integer, an error flag ("_errno") is set to indicate the problem. The term "integer" actually indicates C type "long".

int2str(dst, radix, val) converts the long integer "val" to character form and copies it to the destination string "dst" followed by a terminating null. The result points to the "dst" string, unless the requested radix is out of range, in which case the result is NULL. Digits are generated in reverse order in the appropriate representation, then are placed into the output string. Using "int2str", "itos" and "ltos" are defined.

```
#define itos(x, y) int2str(y, 10, (long)x)
#define ltos(x, y) int2str(y, 10, x)

    char *int2str(dst, radix, val)
    char *dst;
    int radix;
    long val;
    {
        static char _dig_vec[] =
            "0123456789abcdefghijklmnopqrstuvwxyz";
        char buffer[33];
        char *p = &buffer[32];

        if (val < 0)
        {
            *dst++ = '-';
            val = -val;
        }
        *dst = *p = '0';
        if (radix > 36 || radix < 2)
            return NULL;
        do
            *--p = _dig_vec[val%radix];
        while (val /= radix);
        while (*dst++ = *p++);
        return dst-1;
    }
```

str2int(src, radix, lower, upper, &val) converts the string pointed to by "src" to an integer with radix "radix" and stores at "val". Its value usually is a pointer to the next character after the last digit of the number converted. In case of conversion error, "_errno" is set to a nonzero value, the function value is set to NULL, and "val" is set to zero. Using "str2int", "atoi" and "atol" are defined.

```
#define MaxInt      0x7fffL
#define MinInt      0x8000L
#define MaxLong 0x7fffffffL
#define MinLong 0x80000000L
#define EDOM     1
#define ERANGE   2
int _errno;
char _c2type[129] =
    {
      37,      /* EOF == -1 */
      37, 37, 37, 37, 37, 37, 37, 37,
      37, 38, 39, 39, 39, 39, 37, 37,
      37, 37, 37, 37, 37, 37, 37, 37,
      37, 37, 37, 37, 37, 37, 37, 37,
      38, 36, 36, 36, 36, 36, 36, 36,
      36, 36, 36, 36, 36, 36, 36, 36,
       0,  1,  2,  3,  4,  5,  6,  7,
       8,  9, 36, 36, 36, 36, 36, 36,
      36, 10, 11, 12, 13, 14, 15, 16,
      17, 18, 19, 20, 21, 22, 23, 24,
      25, 26, 27, 28, 29, 30, 31, 32,
      33, 34, 35, 36, 36, 36, 36, 36,
      36, 10, 11, 12, 13, 14, 15, 16,
      17, 18, 19, 20, 21, 22, 23, 24,
      25, 26, 27, 28, 29, 30, 31, 32,
      33, 34, 35, 36, 36, 36, 36, 36
  };

char *str2int(src, radix, lower, upper, val)
char *src;
int radix;
long lower, upper, *val;
{
    char *answer;
    int d, n, sign;
    long limit, scale, sofar;
    *val = 0;
    if (radix < 2 || radix > 36)
    {
        errno = EDOM;
        return NULL;
    }
    if ((limit = lower) > 0) limit = -limit;
    if ((scale = upper) > 0) scale = -scale;
    if (scale < limit) limit = scale;
    while (*src == ' ' || *src == 't') src++;
    sign = -1;
    if (*src == '+') src++; else
    if (*src == '-') src++, sign = 1;
    if (_c2type[1+*src] >= radix)
    {
        errno = EDOM;
        return NULL;
    }
    while (*src == '0') src++;
    for (n = 0; _c2type[1+*src++] < radix; n++);
    answer = --src;
    for (sofar = 0, scale = -1; --n >= 0; )
    {
        d = _c2type[1+*--src];
        if (-d < limit)
        {
            errno = ERANGE;
            return NULL;
        }
        limit = (limit+d)/radix;
        sofar += d*scale;
        if (n != 0) scale *= radix;


      }
      if (sign < 0 && sofar < -MaxLong ||
          (sofar*-sign) < lower ||
          sofar > upper)
      {
          errno = ERANGE;
          return NULL;
      }
      *val = sofar;
      errno = 0;
      return answer;
  }

  int atoi(src)
  char *src;
  {
      long val;
      str2int(src, 10, MinInt, MaxInt, &val);
      return (int)val;
  }

  long atol(src)
  char *src;
  {
      long val;
      str2int(src, 10, MinLong, MaxLong, &val);
      return val;
  }
```

The "strfind" and "strrepl" functions find and insert character patterns in strings. They share the use of the "_strpat" function and several auxiliary variable areas. "_strpat" is based upon R. Nigel Hospool's algorithm, as described in "Software Practice and Experience, 1980", page 505. Because the Full C compilers on the 6809 do not implement "unsigned char", the characters are limited to values 0 to 127, the ASCII code sequence.

```
#define _AlphabetSize  128
int      _pat_lim;
int      _pat_vec[_AlphabetSize+1];
static  char  *oldPat = "";
char *_str2pat(pat)
char *pat;
{
    int l, i;
    if (pat == NULL)
        pat = oldPat;
    else
        oldPat = pat;
    for (l = 0; *pat++; l++);
    for (i = _AlphabetSize; --i >= 0;
        _pat_vec[i] = l);
    for (pat = oldPat, i = l; --i > 0;
        _pat_vec[*pat++] = i);
    _pat_lim = --l;
    return oldPat;
}
```

strfind(src, pat) returns a pointer to the first occurrence of "pat" in "src", or returns NULL.

```
char *strfind(src, pat)
char *src, *pat;
{
    char *s, *p;
    int c, lastch;
    pat = _str2pat(pat);
    if (_pat_lim < 0)
    {
        for (s = src; *s++; );
        return s-1;
    }
    for (lastch = pat[c = _pat_lim]; ;
        c = _pat_vec[c])
    {
        for (s = src; --c >= 0; )
            if (!*s++) return NULL;
        c = *s;
        src = s;
        if (c == lastch)
        {
            for (s -= _pat_lim, p = pat; *p; )
                if (*s++ != *p++) goto not_yet;
            return s;
not_yet:;
        }
    }
}
```

strrepl(dst, src, pat, rep, times) copies "src" to "dst", replacing the first "times" non-overlapping instances of "pat" by "rep". It returns a pointer to the null terminating character of "dst".

```
char *strrepl(dst, src, pat, rep, times)
char *dst, *src, *pat, *rep;
int times;
{
    char *s, *p;
    int c, lastch;
    pat = _str2pat(pat);
    if (times <= 0)
    {
        for (p = dst, s = src; *p++ = *s++; );
        return p-1;
    }
    if (_pat_lim < 0)
    {
        for (p = dst, s = src; *p++ = *s++; );
        for (--p, s = rep; *p++ = *s++; );
        return p-1;
    }
    lastch = pat[c = _pat_lim];
    for (;;)
    {


       for (s = src, p = dst; --c >= 0; )
           if (!(*p++ = *s++)) return p-1;
       c = *s;
       src = s;
       dst = p;
       if (c == lastch)
       {
           for (s -= _pat_lim, p = pat; *p; )
               if (*s++ != *p++) goto not_yet;
           for (p = dst-_pat_lim, s = rep;
               *p++ = *s++; );
           --p;
           if (!--times)
           {
               for (s = src; *p++ = *++s; );
               return p-1;
           }
           dst = p;
           src++;
           c = _pat_lim;
       }
       else
       {
not_yet:       c = _pat_vec[c];
       }
    }
}
```

The translate family of functions ("memtrans", "strntran", and "strtrans") translate characters from one string to another according to the contents of "from" and "to" control strings. They use a common routine "_str2map.c" and several common variable areas to construct control tables for the translation process. This routine may be also used for the purpose of constructing the mapping tables separately from the translation routines. When the translation routines are called, a NULL for the "from" or "to" parameters indicates to use the table previously defined. Because the Full C compilers on the 6809 do not implement "unsigned char", the characters are limited to values 0 to 127, the ASCII code sequence.

```
#define _AlphabetSize  128
static  char  *oldFrom = "?";
static  char  *oldTo   = "?";
char    _map_vec[_AlphabetSize+1];

_str2map(option, from, to)
int option;
char *from, *to;
{
    int i, c;
    if (from == NULL && to == NULL) return;
    if (from == NULL)
        from = oldFrom;
    else

        oldFrom = from;
    if (to == NULL)
        to = oldTo;
    else
        oldTo = to;
    switch (option)
    {
        case 0:
            for (i = _AlphabetSize; --i >= 0;
                _map_vec[i] = i);
        case 1:
            while (i = *from++)
            {
                _map_vec[i] = *to++;
                if (!*to)
                {
                    c = *--to;
                    while (i = *from++)
                        _map_vec[i] = c;
                    return;
                }
            }
            return;
        case 2:
            c = *to;
            for (i = _AlphabetSize; --i >= 0;
                _map_vec[i] = c);
            while (c = *from++)
                _map_vec[c] = c;
            return;
    }
}
```

memtrans(dst, src, from, to, len) copies "len" characters from "src" to "dst", translating characters in "from" to corresponding characters in "to".

```
memtrans(dst, src, from, to, len)
char *dst, *src, *from, *to;
int len;
{
    _str2map(0, from, to);
    while (--len >= 0) *dst++ = _map_vec[*src++];
}
```

strntran(dst, src, len, from, to) copies up to "len" characters from "src" to "dst", translating characters in "from" to "to". It fills in the remainder of "dst" up to "len" bytes with null characters.

```
strntran(dst, src, len, from, to)
char *dst, *src, *from, *to;
int len;
{

    _str2map(0, from, to);
    while (--len >= 0 &&
        (*dst++ = _map_vec[*src++]));
    while (--len >= 0) *dst++ = '\0';
}
```

strtrans(dst, src, from, to) copies characters from "src" to "dst", translating characters in "from" to "to".

```
strtrans(dst, src, from, to)
char *dst, *src, *from, *to;
{
        _str2map(0, from, to);
        while (*dst++ = _map_vec[*src++]);
    }
```

Next month's column will cover several applications of O'Keefe's functions and will continue the discussion of string-handling functions in the C language.

C PROBLEM

The following program:

```
    #include "stdio.h"
    #define exp(x) if ((x) == '\t') printf(" ")
    main()
    {
        char c[] = "abc\tdef";
        char *p;
        for (p = c; *p; p++)
        {
            if (*p != 'c')
                exp(*p);
            else
                printf("%c",*p);
        }
    }
```

outputs the following:

```
    ab def
```

This is because the unmatched "if" in the definition matches the "else", despite what the indentation seems to imply, and the 'c' is not output.

The guideline illustrated by this problem is to code complete expressions or statements in definitions, to avoid such unexpected matchups.

The next problem is to write a program which translates the upper case letters in a file to lower case and drops all control characters (except carriage return) using the appropriate translate functions described earlier in this article.

EXAMPLE C PROGRAM

Following is this month's example C function; it is from Phil Gunsul, and provides an INTROL C program to convert a FLEX binary file to S1 format.

```
    #include        "stdio.h"
    #include        "flex.h"
    #define WIDTH 32
    char    count, buff[WIDTH];
    int     curadd, address, sladd;

    main(argc, argv)
    int     argc;
    char    *argv[];
    {
            FILE    * fpin;
            int     i, byte;
            char *add;

        if (argc != 2) {
                printf("Syntax:\thexdmp filename\n");
                exit(1);
        }
        if ((fpin = fopen(*++argv, "br")) == ERROR) {
                printf("Cannot open input file.\n");
                exit(1);
        }
        i = 0;
        add = &address;
        curadd = 0xffff;
        while ((byte = getc(fpin)) != ERROR) {
                if (byte == 2) {
                        add[0] = getc(fpin);
                        add[1] = getc(fpin);
                        if ((count = getc(fpin)) == 0)
                                break;
                        if (curadd != address) {
                                curadd = address;
                                header(i);
                                i = 0;
                        }
                        while (1) {
                                buff[i++] = getc(fpin);
                                curadd++;
                                if (i == WIDTH) {
                                        header(i);
                                        i = 0;
                                }
                                if (--count == 0)
                                        break;
                        }
                }
                if (byte == 0x16) {
                        getc(fpin);
                        getc(fpin);
                }
        }
        header(i);
        printf("\nS9\n");
        exit(0);
}


int size;
{
        char    cnt, chksum, *add;
        int i;

        if (size != 0) {
                add = &sladd;
                if (size > WIDTH)
                        cnt = WIDTH;
                else
                        cnt = size;
                chksum = size + add[0] + add[1];
                printf("\nS1%02X%04X", cnt + 3, sladd);
                for (i = 0; i < cnt; i++) {
                        printf("%02X", buff[i]);
                        chksum += buff[i];
                }
                printf("%02X", ~chksum - 3);
        }
        sladd = curadd;
}
```

68000 USER NOTES

Philip Lucido
2320 Saratoga Drive
Sharpville, PA 16150

Last month concerned the writing of programs, mostly in the C language, so that they could be easily transported between operating systems, particularly between 6809 and 68000 versions of OS-9. Mostly I dealt with trouble spots to watch out for, such as differences in compilers, libraries, and data types. I would now like to continue, covering techniques I have actually used or have seen elsewhere.

As the second part of this month's column, I have the beginning of some information on Motorola's new 32 bit microprocessor, the 68020, with more to follow next month.

Data Types (again)

One of the most troublesome problems encountered in porting programs between 8 bit and 16 bit computers has to do with the changing sizes of the basic data types. I mentioned this last month, and said that a program should never make rash assumptions about the size of a variable, but instead should use the sizeof function. This only t kes care of one side of the problem, though. What should you do, for instance, when you want a variable to be two bytes long, regardless of the operating system? This might happen when using a file which has a fixed structure which must be exactly duplicated in all programming environments.

Here is where the pre-processor again steps in. Generally, it is possible to find data types which have a given size on various machines. The only problem is that the type might differ between machines. Instead of using the name of one of the primitive data types, then, some standardized set of special names, each of which is to have a particular size and range of legal values on all machines, is defined. As an example:

```
    #define BYTE char
    #define WORD short
    #define LWORD long
    #define UBYTE unsigned char
    #define UWORD unsigned short
    #define ULWORD unsigned long

    BYTE tinyval;
    WORD smallval;
    LWORD bigval;
```

By using the defined names, instead of the primitive types, a program is protected from changing data type sizes. As it happens, these definitions will work for both the 6809 and 68000, but that is not guaranteed for all processors. Also, it still makes sense to use the defined names, since they tend to make clearer within a program the fact that a variable has been constrained to a particular size.

To encourage the use of special names such as these, it makes sense to place the definitions in a file to be referenced by an **#include** statement. For instance, I have taken to placing these definitions in a file named stddefs.h, placed in the same directory as stdio.h, so that the reference in every C program is **#include <stddefs.h>**. I chose not to place the definitions in stdio.h, as might seem logical, since I would then have to edit stdio.h with each new release of the compiler.

Assuming that each operating system uses a different disk (or disk partition) for the standard include files, separate stddefs.h files can be created, each one being particular to a single operating system or machine. On the other hand, the same file can be used in all cases, with any required differences being implemente using #ifdef/#else/#endif constructs, as discussed last month.

There is an alternate method for declaring these special names, via the **typedef** specifi r. Instead of

```
    #define BYTE char
```

the line could be

```
    typedef char BYTE;
```

Which form is used is probably just a matter of personal preference, though the fact that the #define is handled by the pre-processor, while the typedef is handled by the compiler proper can make some difference (see K & R, page 141, on typedefs for an example).

There is a final point to be made here in conjunction with 4 byte long variables and printf. Since **long** and **int** are synonymous on the 68000, it is never necessary to use the 'l' specification in printf control strings (e.g. "%8d" instead of "%8ld"). Use printf on a long variable on the 6809 without the 'l', though, and everything will fall apart, since printf has no way of knowing that the variable is actually taking up 4 bytes on the stack. To defend against this, always use a special name, such as the LWORD defined above, for values which require more than 2 bytes, and always use the long descriptor 'l' in printf cont ol strings for such variables.

Getting the Most From Your Compiler

A compiler for a particular machine will generally have some ability to create more efficient programs. For instance, 68000 C compilers will generally be able to allocate a large number of register variables, perhaps 6 or more, while 6809 C compilers only allow one. Microware's 6809 C, on the other hand, has a type class specifier of **direct** for global variables which should be referenced using direct page instructions if at all possible, instead of using longer indexed offset instructions.

The large number of register variables available with the 68000 would seem to indicate that everything in sight should be declared register if possible. If this is done, though, the code produced on the 6809 may be less efficient than possible. Consider a function with one or more arguments, like main(argc,argv). Both argc and argv should likely be declared register on the 68000. However, it may be better on the 6809 to save the only register declaration for a local variable within the function.

Again, it makes sense with the 6809 to use the **direct** specifier. Most other compilers will error out when the **direct** is encountered, though (as it happens, the Microware 68000 C compiler just ignores it).

How can portable programs use those special efficiency features which may be available? As you might expect, the pre-processor is used. The following lines occur in my stddefs.h file:

```
    #ifdef OSK      /* 68000 */
    #       define REGISTER register
    #       define DIRECT
    #else             /* 6809 */
    #       define REGISTER
    #       define DIRECT direct
    #endif
```

When I wish to defer using the single register declaration for the 6809, but still wish to use the 68000 register variables, I just use the REGISTER declaration instead:

```
    main(argc,argv)
    REGISTER int argc;
    REGISTER char **argv;

    {   REGISTER int vall;
        register char *p;
```

For the 68000, all of the variables will be placed in registers. For the 6809, REGISTER translates to nothing, so that only the variable 'p' is placed in a register. In the same way, any global variables can be declared DIRECT, but the declaration will only take effect on the 6809.

68020

Motorola was a little slow out of the gate with the 68000, trailing the release of Intel's 8086 by a couple of years. We all know what that lead to, with the IBM PC and various PC compatibles every here you look. It is kind of nice, then to see Motorola's 32 bit micro, the 68020, make it out so early as compar d to other 32 bit designs.

I just received the 68020 user's manual, which describes the chip from both a hardware and software viewpoint. For those of you who are interested, the book is the **MC68020 32-Bit Microprocessor User's Manual**. It is published by Prentice-Hall, and has a part number of MC68020UM(AD1). I'm not sure if that is a Motorola part number or the Prentice-Hall number. You can probably get it through most bookstores, though I got mine via an electronics distributor my company uses. Distributors, like Pioneer, Hamilton-Avnet, or Arrow, may be able to get the book to you more quickly.

Since the book came in very recently, I haven't had time to peruse it carefully, so all I can say now are some general first impressions. I'll be more detailed next month. For now, I will only cover a few software aspects.

The 68020 is upwards compatible, at the object code level, with the previous members of the family (68000, 68008, 68010, and 68012). There are three types of software extensions: new instructions, extended instructions, and new addressing modes.

The new instructions include two major groups of new instructions. The first group deals with a new data type, bit fields. A bit field is a string of consecutive bits, without regard to actual byte boundaries. A single bit field can be from 1 to 32 bits long, and can be offset from a base byte boundary by -2^{31} to $(2^{31})-1$ bits. Bit fields can be extracted, being moved right justified into a data register, with or without sign extension, as well as moved in the other direction, an insert. The other bit fields opcodes test the value of a bit field (compare with 0), set, clear, or complement a bit field, and find the first bit which is set within a bit field. These opcodes seem to be explicitly defined for use in implementing the C languag 's bit fields, as well as the set data type in Pascal.

The other group of new instructions consists of coprocessor communication instructions. These instructions create a method in the 68020 for interfacing with other intelligent devices which greatly extend the 68020's power. Current coprocessors include a floating point chip and a virtual memory management unit. One instruction allows access to any of the predefined coprocessor instructions. Other instructions, analogous to the 68000 instructions Scc, DBcc, Bcc, and TRAPcc (a new conditional TRAP instruction), allow processing to depend on a coprocessor's current state. There are also instructions for saving and restoring a coprocessor's current state.

Certain 68000 instructions have been extend d in the 68020. For instance, the relative branch instructions BRA, BSR, and Bcc now accept 32 bit offsets, so subroutine calls are no longer limited to a 64K range. A multiply with a 64 bit result, as well as a 64 bit/32 bit divide, is now available. As mentioned above, there is a new conditional form of the TRAP instruction. There is also a new sign extend instruction, EXTB, which extends a byte value directly to a long value. This also seems to be motivated by C, since current implementations require many EXT.W, EXT.L sequences when dealing with yte variables.

There are a number of new addressing modes available. Constant offsets of 32 bits are now allowed. Index registers, in modes like (d8,An,Xn), can now be specified with a scale factor of 1, 2, 4, or 8. This means that the value in the index register will be shifted before adding, with no extra clock cycles, so that indexed offsets into word, long word, and quad word arrays can be done without using explicit shift instructions. Finally, there is a very general indexing form, which can involve two constant offsets, two registers, and memory indirection, with every element optional. As an example, an addressing mode of ([1000,A5],D1*4) says t add 1000 to the value in A5, giving an address in memory. The 4 byte pointer at this address is retrieved, and 4 times the value in D1 is added, giving a final address. This instruction might conceivably appear in a C pr gram in which a pointer variable (at 1000,A5) points to an array of long integers.

There is much, much more to cover (for instance, the 68020 comes in a quad pin array package, with over 110 pins!) but I have already delayed this c lumn too long waiting for the 68020 manual to appear. I will certainly have more to say next month, once I have a chance to read the book more closely. In the meantime, send for the manual yourself, and follow along next month.

- - -

TMP/FREEFORM FILER

Reviewed by Bob Nay

The TMP™/FreeForm Filer Package is a Free Form, or "Unstructured", File Management System offered by **The United Software Company**, 2431 E. Douglas, Wichita, KN 67211 (316) 684-5281 (TMP is a Trademark of The United Software Company). Unlike most File Management and Data Base Management Systems, **TMP/FreeForm Filer** does not require the definition of specific 'Fields' to make up a "structured" Record System.

The "TMP" in the Package name refers to United Software's Total Management Planning system of Software Packages, which includes the **TMP/Front-End** (a Menu-Controlled "System Manager" Front End Package), the **TMP/Manager** I (a "structured" Data Base System), **TMP/PowerPlanner** (renamed from TMP/Calc - an Electronic Spreadsheet Program), plus others, which are all designed to be fully "data compatible" and provide a common User Interface and Command Structure.

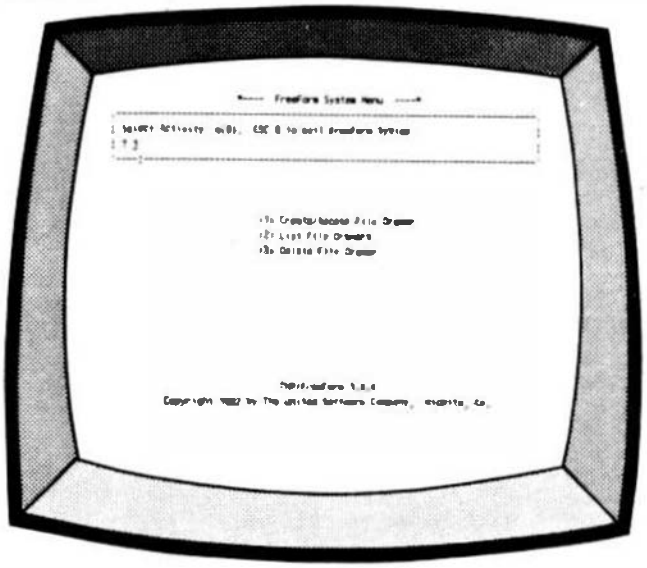
TMP/FreeForm Filer for OS-9 sells for $225.00 and requires OS-9 **Level II** and at least **128K** of Memory. You have seen it advertised with t e **Smoke Signals** Computer Systems in the past, and **United Software** is now making it available for other Systems. We understand that it also runs under MS-DOS and some other Operating Systems at this time, and that FLEX and UniFLEX versions are being considered.

In order to visualize the concept of **TMP/FreeForm Filer**, imagine a 3x5 Card File Drawer full of Cards of information. **TMP/FreeForm Filer** was designed to allow the User to rapidly locate a specific item of information that is in these cards through searching for certain Card **Names** and/or one of the specified **Key Words** on any of the Cards.

TMP/FreeForm Filer allows as many "Drawers" (Data Files) as you have Disk Storage for. Each "Drawer" may contain up to 32,000+ "Cards", and each "Card" can contain up to 9 "Pages" of 37 Columns by 13 Rows of information. Each "Page" may contain up to 13 20-character "Key Words" for subject searches, etc. (providing a maximum of 117 "Key Words" per "Card"). The Data is maintained in two Files; a Data File and a Key File (which are generated and maintained automatically by the Program). Searches for "Key Words" and/or "Card Titles" allow 'wild card' selection with the "*" (match ANY NUMBER of characters) and the "?" (match any SINGLE character), but **TMP/FreeForm Filer** does NOT provide any combinational selections such as 'this AND this' or 'this OR that'. You **CAN** get a two-level selection by using both a Card Title **AND** a Key Word selection Mask (and proper selection of Card Titles can be a big help here). You can examine selected Cards of information on the Display, or output them to the Printer or to another File (in an ASCII Format suitable for use with a Word Processor or Editor) with **TMP/FreeForm Filer**, but you can only output in a 3x5 format suitable for printing on single-wide 3x5 continuous form cards (available from Computer Supply houses) or in an 8 1/2 x 11 paper format (which separates each card page with a line of asterisks).

The **TMP/FreeForm Filer** Disk contains several Files including the Command Files, an Environment Editor for setting the Program up for your specific Terminal (with "TMP.ENV" Files for several Terminals already set up), the "Help" Files, and an Example File set. Two Files, "FREEFORM" and "TMPENV", must be copied to you Execution Directory. Three Files, "HELPFF.DHP", "HELPFF.KHP", and the "TMP.ENV" File for your specific Terminal (made up with the "TMPENV" Command if your Terminal is not already provided) must be in the Data Directory which contains the "Drawer", or Data Base, that you will be working with.

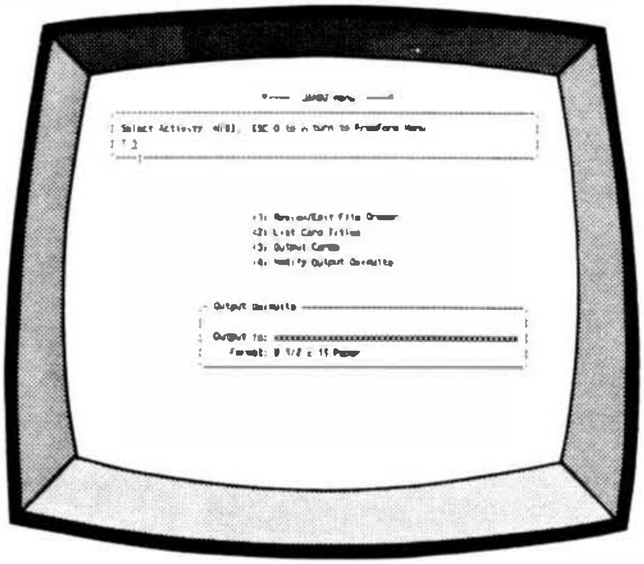
Overall System Operation consists of calling up the Program with the command "**freeform**", which produces a "System Menu" (Fig. 1) which provides the access and overall control of the various "Drawers", or Data Bases, that may be in the present Data Directory. The "System Menu" allows the Listing of the Drawers (like a "Dir"), the Deletion of Drawers (delete a complete Data Base), and the Creation or Access of any Drawer through the selection of a Menu option Number.



```
---- Freeform System Menu ----

Select Activity (n)?, ESC Q to exit Freeform System
? 2

1) Create/Access File Drawer
2) List File Drawers
3) Delete File Drawer

TMP/FreeForm 1.x
Copyright 1982 by The United Software Company, Wichita, Ks.
```

Fig. 1 -- Freeform System Menu

Should a Drawer be chosen for Creation or Access, the Program moves on to the next Menu, which is the "File Drawer" Menu (Fig. 2). From this Menu, the User has access to overall Card File manipulations, such as Listing the Card Titles (i.e., a "Dir" of the Card Names that are in the Drawer chosen from the last Menu), Outputting all or a selection of the Cards to a Printer or File, changing the Output Defaults (changing the Printer name or specifying a Filename, and toggling the output Format between a 3x5 Card Format or an 8 1/2 x 11 Paper Page Format), or moving on to the "Freeform Data Entry" Screen for operations on the Individual Cards themselves. Again, the Menu is operated through the selection of an option Number.



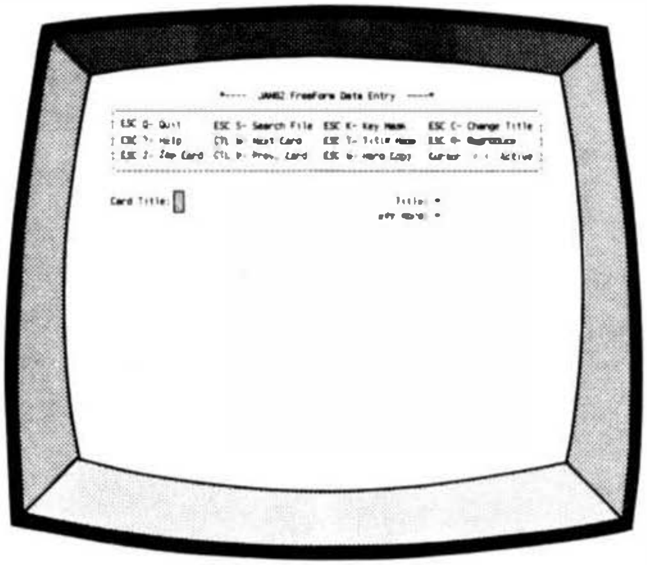
```
*---- JAN82 Menu ----*
! Select Activity (#1-4), ESC Q to return to Freeform Menu    !
! ? 3                                                         !

        (1) Review/Edit File Drawer
        (2) List Card Titles
        (3) Output Cards
        (4) Modify Output Defaults

     - Output Defaults ------------------------------------
     !                                                    !
     ! Output to: [illegible]                             !
     !    Format: 8 1/2 x 11 Paper                        !
```

Fig. 2 -- Freeform File Drawer Menu,
showing the Drawer (Data Base)
named "JAN82" selected

The "Freeform Data Entry" Screen provides a two-part access to working with the Cards; first is the Card Selection operations, and then the actual Card Manipulation operations. Selection of the desired operations at this level in the Program is through "ESC <letter>" and "Control Key" operations.

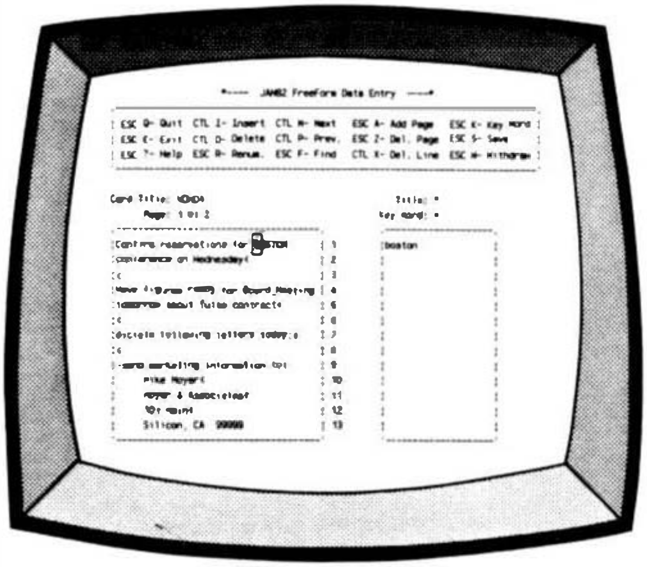
On initial entry into the "Freeform Data Entry" display (**Fig. 3**), an 'Available Commands' block at the top of the Display provides a series of 11 Escape and Control Key operations that allow the User to select the specific Card that is to be worked on. Throughout the Program, an 'ESC Q" sequence Quits the present level of operation and moves the User back a level, or Menu, 'house cleaning' as it goes. In both levels of the "Freeform Data Entry" display, the User can "Quit" or get "Help" relevant to the level he is in. From the Card Selection level, the User can specify a Card Name, or locate a Card through either a partial-name "Search" or through a Title (Card Name) AND/OR Key Word "Mask", using either a full character Mask (i.e., giving the full Title or Key Word), or through a Wild-Card Mask using the "*" and/or "?". Once a Card is located, it can be "Zapped" (deleted from the File), its Title can be changed, it can be output through the "Output Defaults" specified in the previous Menu, it can be "Reproduced" onto new Cards as often as you want (this feature is provided to make the use of "Templates" easy), or the User can move to the "Previous" or "Next" Card in the File. All Cards are maintained alphabetically by Card Title, so if a Search or Mask has located a Card close to the Card desired, this procedure provides an easy way to locate the desired information.

Finally, once a Card is located, a total of 15 Escape and Control Key operations allow the examination, editing, or otherwise "massaging" the information that is on the Card just located (see **Fig. 4** and **5**). As before, the "Quit" and "Help" commands are available; the "Quit" command saves any changes made to the Card, and the "Help" info is relevant to this level of the Program. An 'Exit" is provided to leave without changing anything. The User can insert information, Delete a character, line, or any of the 9 pages in the Card, step to the Next or Previous pages, Find a specific Page, change the order of the Pages by Renumbering them, Add Pages, and specify the "Key Words" in the information that are to be used for locating that specific Card. Finally, a "Save" and "Withdraw" are provided which saves a page of information to an internal buffer, which can be withdrawn onto any page at any time. The information saved to the buffer remains there as long as the Program is active, or the buffer is overwritten with another Save, and any information that may have been on a page when the buffer is withdrawn onto that page is completely replaced by the new information. Where the "Reproduce" command in the previous level produced NEW Cards, the Withdraw command at this level simply REPLACES any information that may have been on a page. Again, the primary purpose of this feature is for "Templating" a page of information.



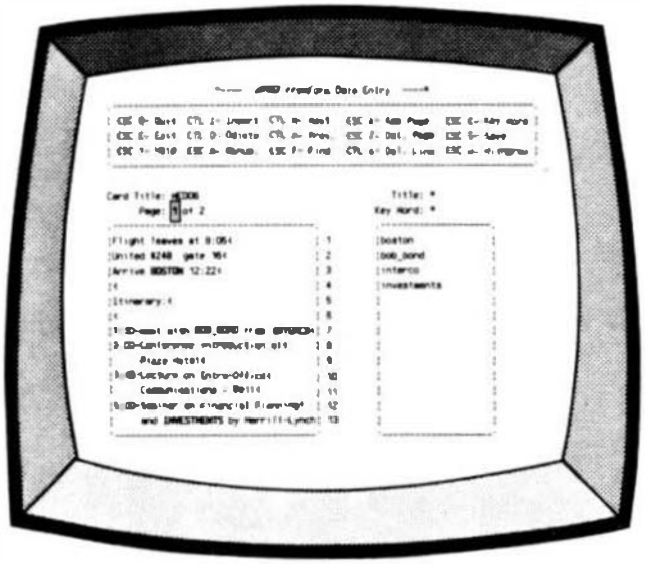
```
*---- JAN82 Freeform Data Entry ----*
! ESC Q- Quit      ESC S- Search File  ESC K- Key Mask    ESC C- Change Title !
! ESC ?- Help      CTL N- Next Card    ESC T- Title Mask  ESC R- Reproduce    !
! ESC Z- Zap Card  CTL P- Prev. Card   ESC H- Hard Copy   Cursor [illegible] Active !

Card Title: █                              Title: *
                                        Key Word: *
```

Fig. 3 -- The initial Freeform Data Entry screen



```
*---- JAN82 Freeform Data Entry ----*
! ESC Q- Quit  CTL I- Insert  CTL N- Next   ESC A- Add Page   ESC K- Key Word !
! ESC E- Exit  CTL D- Delete  CTL P- Prev.  ESC Z- Del. Page  ESC S- Save     !
! ESC ?- Help  ESC R- Renum.  ESC F- Find   CTL X- Del. Line  ESC W- Withdraw !

Card Title: [illegible]                     Title: *
      Page: 1 of 2                       Key Word: *
---------------------------------------         -----------------
!Confirm reservations for BOSTON        ! 1     !boston          !
!conference on Wednesday!               ! 2     !                !
!(                                      ! 3     !                !
!Have figures ready for Board_Meeting   ! 4     !                !
!tomorrow about future contracts        ! 5     !                !
!(                                      ! 6     !                !
!Dictate following letters today:(      ! 7     !                !
!(                                      ! 8     !                !
!Send marketing information to:         ! 9     !                !
!     Mike Meyers                       ! 10    !                !
!     Meyer & Associates                ! 11    !                !
!     101 Main                          ! 12    !                !
!     Silicon, CA  99999                ! 13    !                !
---------------------------------------         -----------------
```

Fig. 4 -- Freeform Data Entry screen
showing Page 1 of a 2-page Card.
"boston" has just been selected as a Key Word.
Notice the "Free Form" entry of information
in this and Fig. 5's screens.



```
Freeform Data Entry

Card Title: [illegible]              Title: *
      Page: 1 of 2                Key Word: *

Flight leaves at 8:05         1   boston
United #248  gate 16          2   bob_bond
Arrive BOSTON 12:22           3   interco
                              4   investments
Itinerary:                    5
```

Fig. 5 -- Freeform Data Entry Screen showing 4 words defined as Key Words and Cursor in position to "Renumber" the Page Number.

In summary, calling up the Program produces a "System Menu" (Fig. 1) which allows the specification and control of the Drawers, or Data Bases, themselves. The flow moves from the "System" level to a "File Drawer" Menu (Fig. 2) for the Drawer selected, which provides overall control of the full "Deck of Cards". From the "File Drawer" Menu, the Program moves on to the "Freeform Data Entry" display, which provides an initial level of control for locating and/or manipulating a complete Card of Information (Fig. 3). Once a specific Card has been selected, control moves to the second level of the "Freeform Data Entry" (Figures 4 and 5), which allows the control and manipulation of the pages and information that is contained within the Card itself.

TMP/FreeForm Filer is a flexible Program that can be used for a multitude of tasks without requiring a "Computer Science" background to understand how to set the system up for your information. As with any piece of Software, the more you work with it, the more ideas for using it appear. The Menus may not appear too "palatable" to some users, but they do not interfere very much with the normal operation for the experienced Computer User (most of your actual working time is spent in the "Freeform Data Entry" Screen, where the more familiar ESC and CTL Key Commands are used), while isolating the Operating System for the newer Users.

The **TMP/FreeForm Filer** Documentation is several notches ABOVE what the SS-50 Bus Community is used to seeing. The first part of the Manual discusses the overall TMP System User Interface and includes a detailed Chapter on setting the Program up for your specific Terminal and preferences, while the second Section discusses **TMP/FreeForm Filer** Operation under OS-9. This section begins by discussing a "Real World Example" in general (in the Manual, they use a simple Appointment Calendar for their example), and then walks the new User through the use of the Program in setting up and working with this example. Finally, there are detailed chapters on the Menus and Commands, and a chapter on Interfacing **FreeForm Filer** with other TMP Packages. The Manual ends with a few pages of "Some Ideas", an Appendix of "Basic System Rules", an Appendix of "Errors", a Glossary, and an Index. The Manual is well laid out and easy to read and use (the Figures provided above came directly out of the Manual).

Since the Package is designed for quick access to a specific piece of information, I found it a little cumbersome to have to work through the Menus just to get one item, but I have been informed by **United Software** that the Disk now contains a Program called **"QFind"** which allows the User to locate a specific item directly from the Operating System. Another feature that would radically enhance the use of this type of Data Information System would be the capability of multi-level searches, such as "this AND that" or "this OR that but NOT such-and-such". This would allow looking for something like "high temp AND small spots AND enlarged glands" in a Drawer of Childhood Diseases, or "tall AND red OR yellow AND annual" in a Drawer of Flowers. Again, **United Software** indicates that that feature is on the "Things to Do" list, and will probably be provided as an additional module for **TMP/FreeForm Filer** at a later date.

As it now stands, some thought in naming Cards and Key Words, along with the use of the Wild Card Mask capabilities, can provide similar capabilities. For example, a Card in a Drawer of "Flowers" might be named "TallRedAnnual" (up to 20 characters can be used for both Card Names and Key Words, and the "underline" character can be used to tie words together if desired), where Card Name Masks of "*red*", "tall*", and "*annual" would all locate this Card. The same concepts can be used with Key Words. Another possibility would be to use a set of Cards with somewhat similar Names that would provide "General" subject categories which would contain the names of other Cards and/or Key Words that contained more detailed information. The first search would provide "pointers" to more specific information. The Program as it now stands is being used by Medical and Dental Researchers, so it has a lot of capability. Again, a little thought can provide unlimited possibilities.

Finally, I hit ONE snag with the Program; the "Help" Files would generate an "Error 198" on the Gimix III System, which immediately "bombs" you out of the Program and back to the Operating System. This is a HARDWARE Trap in the Gimix III System that indicates that the Program is trying to change the Program Memory (which is a theoretical No-No in OS-9 -- only DATA Areas can be changed!). **TMP/FreeForm Filer** is not unique in having this problem; I have run into it often with the Gimix III System. Gimix installed this Trap in the System III Computers to protect one User from another Users Program, but it sure would be nice to have a Switch to Enable/Disable this feature. Compilers, especially, have problems in this area. Anyway, **United Software** is looking into it, and the overall Program operation is so simple that the "Help" Files are seldom needed (and the Manual is so well laid out that it is easy to find the information when needed in IT).

All in all, TMP/FreeForm Filer is a very useful Program, and it is hoped that it can be "ported" over to OS-9 Level I and the FLEX System (it is written in C, which makes it transportable — THEORETICALLY!!!).

MAJOR ENHANCEMENT OF SWPTC MIRROR.CMD

Enclosed is a copy of a rewrite and major enhancement of the SWTPC FLEX 2.0:3 program named MIRROR. This program uses all of the memory available to perform a very fast track-oriented copy of one diskette to another. It reads and writes the sectors in a manner designed to require the least amount of time to process all sectors on a track. A double-sided, double-density eight-inch diskette may be copied to another in about 1.5 minutes, assuming the SWTPC FLEX interlace is used on both diskettes.

Since this program ignores the logical organization of the data on a diskette, the diskettes must have compatible formats. This implies that they must have the same number of sectors per track, neither may contain any defective sectors, and the number of tracks on the input diskette must not be greater than the number of tracks on the output diskette. The interlaces on the diskettes are not required to be identical; however, minor to severe degradation may be experienced on nonstandard interlaces. Diskettes created by 6800 FLEX systems may not be MIRRORed at all, since they lack sector 01 on track 00.

The major improvements in this program are as follows:

1. It issues a confirming prompt before attempting to read either diskette, providing a means of using MIRROR without requiring it to be on either input or output diskette.
2. It allows the input and output drive numbers to be the same, in which case it issues prompts to write-protect the input diskette and to indicate when to swap diskettes.
3. It displays the disk volume name, number, and creation date for each input and output diskette, helping to prevent backward copying, which is usually inconvenient and occasionally disastrous.
4. It uses revised wording for most of the prompts to be more helpful to the inexperienced or occasional user.
5. It always performs a verify operation on the output disk regardless of the FLEX VERIFY flag or MIRROR parameters.

Neither the original SWTPC MIRROR nor this version may be used during spooling operations, be used with any other programs which use the FLEX system FCB, nor be used from BASIC, as all of user memory from $0000 to MEMEND is overwritten by MIRROR. Both versions also use an almost-undocumented disk driver entry point, at location $DE1B, called DSEEK, which causes the disk controller to explicitly seek a track/sector combination contained in the D register. Thus MIRROR may not be usable (without modification) on those systems which do not support DSEEK.

Sincerely,

E M Pass

E. M. Pass

```
                    * mirror (single or double disk)
                            opt     pag
                    * lib flexlib not listed
                            opt     lis
                    *
0000                        org     $0000
                    *
0000 20   1E        start   bra     start1
                    *
0002 02 2E 00 3A    vn      fcb     $02,$2e,$00,$3a,$00 2.0:0
0007 20 31 20 6F            fcc     ' 1 or 2 drive duplication'
                    *
0020 17   00D0      start1  lbsr    getdno    get source drive number
0023 F7   C5BA              stb     srcdrv
0026 17   00CA              lbsr    getdno    get target drive number
0029 F7   C5BB              stb     trgdrv
002C F0   C5BA              subb    srcdrv
002F F7   C5BE              stb     single
0032 17   00D4              lbsr    ckdelm    scan rest of line
0035 27   05                beq     disdrv
0037 17   00C9      scanrl  lbsr    nxtchk
003A 26   FB                bne     scanrl
                    *
003C B6   C5BB      disdrv  lda     trgdrv    display drive numbers
003F 8B   30                adda    #$30
0041 B7   01C4              sta     lsctdr
0044 B6   C5BA              lda     srcdrv
0047 8B   30                adda    #$30
0049 B7   01B9              sta     lscsdr
004C 8E   019B              ldx     #lscrat
004F BF   C5B4              stx     lasmsg
0052 17   C27B              lbsr    putget    ask about drive
0055 1026 C2C6              lbne    pabort
```

```
                    *
0059 BD   CD24      chksir  jsr     pcrlf     check target sir
005C F6   C5BB              ldb     trgdrv
005F 17   C299              lbsr    ptarge    insert target disk
0062 17   C21D              lbsr    getsir
0065 1026 0111              lbne    pbdsir
0069 EC   88 66             ldd     $66,x     max track & sector - target
006C 34   06                pshs    d
006E 8E   01F1              ldx     #ltrgnm   target disk name
0071 17   00A0              lbsr    pdinfo
                    *
0074 7D   C5BE      chksi2  tst     single    check for single disk mirror
0077 26   06                bne     chksi3
0079 8E   01C9              ldx     #lwprot   write-protect source disk
007C BD   CD1E              jsr     pstrng
                    *
007F F6   C5BA      chksi3  ldb     srcdrv    check source sir
0082 17   C25C              lbsr    psourc    insert source disk
0085 17   C1FA              lbsr    getsir
0088 1026 00EE              lbne    pbdsir
008C EC   88 66             ldd     $66,x     max track & sector - source
008F FD   C5B8              std     maxtrk
0092 8E   020C              ldx     #lsrcnm   source disk name
0095 17   007C              lbsr    pdinfo

0098 35   06                puls    d         retrieve target track & sector
009A F1   C5B9              cmpb    maxsec
009D 1026 00DF              lbne    pdform
00A1 B1   C5B8              cmpa    maxtrk
00A4 1025 00DE              lblo    psrgtg
                    *
00A8 17   C1F9      chksi4  lbsr    lookup    look up max sector in table
00AB 1026 00DD              lbne    pnonst
00AF BF   C5AA              stx     tabptr
00B2 8E   CB90              ldx     #sysfcb+$50 id info in sir
00B5 108E C5BF              ldy     #idbuff   id hold area
00B9 C6   0D                ldb     #$0d      13 bytes
00BB 17   C1DE              lbsr    movexy    move it
                    *
00BE B6   C5B9      chksi6  lda     maxsec    compute memory required
00C1 C6   03                ldb     #$03
00C3 3D                     mul
00C4 C3   0001              addd    #$0001
00C7 34   06                pshs    d
00C9 FC   CC2B              ldd     memend
00CC A3   E1                subd    ,s+
00CE FD   C5B2              std     loctab
00D1 83   00FF              subd    #$00ff
00D4 C6   FF                ldb     #$ff
00D6 5C             chksi8  incb
00D7 B0   C5B9              suba    maxsec
00DA 24   FA                bhs     chksi8
00DC F7   C5B7              stb     trkfit    number of integral track fits
00DF 1027 00AF              lbeq    pmemry
                    *
00E3 8E   0227      acontn  ldx     #lcontn   continue (y/n]?
00E6 17   C1E7              lbsr    putget
00E9 1026 C232              lbne    pabort
00ED BD   CD24              jsr     pcrlf     crlf
00F0 16   C00D              lbra    setup1    continue
                    *
00F3 BD   CD48      getdno  jsr     indec     get drive number
00F6 25   7C                bcs     pbddrn
00F8 8C   0004              cmpx    #$0004
00FB 24   77                bhs     pbddrn
00FD 5D                     tstb
00FE 27   74                beq     pbddrn
0100 1F   10                tfr     x,d
0102 39                     rts
                    *
0103 BD   CD27      nxtchk  jsr     nxtch     scan next item
0106 B7   CC11              sta     lsttrm
0109 B6   CC11      ckdelm  lda     lsttrm    check scanned item
010C 81   0D                cmpa    #$0d
010E 27   03                beq     ckdelx
```

```
0110 B1   CC02            cmpa   eolchr
0113 39           ckdelx  rts
                  *
0114 BD   CD1E    pdinfo  jsr    pstrng     print disk info

0117 C6   08              ldb    #$08       print disk name
0119 8E   C890            ldx    #sysfcb+$50
011C A6   80      pdinfl  lda    ,x+
011E 34   14              pshs   b,x
0120 BD   CD0F            jsr    outch
0123 35   14              puls   b,x
0125 5A                   decb
0126 26   F4              bne    pdinfl
0128 86   20              lda    #$20
012A BD   CD0F            jsr    outch
012D 86   23              lda    #$23       #
012F BD   CD0F            jsr    outch
0132 5F                   clrb              print volume number
0133 30   03              leax   $03,x
0135 BD   CD39            jsr    outdec
0138 86   20              lda    #$20
013A BD   CD0F            jsr    outch
013D 7F   C5B2            clr    loctab     print date
0140 B6   CBA3            lda    sysfcb+$63
0143 B7   C5B3            sta    loctab+1
0146 8E   C5B2            ldx    #loctab
0149 5F                   clrb
014A BD   CD39            jsr    outdec
014D 86   2F              lda    #$2f       /
014F BD   CD0F            jsr    outch
0152 B6   CBA4            lda    sysfcb+$64
0155 B7   C5B3            sta    loctab+1
0158 8E   C5B2            ldx    #loctab
015B 5F                   clrb
015C BD   CD39            jsr    outdec
015F 86   2F              lda    #$2f       /
0161 BD   CD0F            jsr    outch
0164 B6   CBA5            lda    sysfcb+$65
0167 B7   C5B3            sta    loctab+1
016A 8E   C5B2            ldx    #loctab
016D 5F                   clrb
016E BD   CD39            jsr    outdec
0171 7E   CD24            jmp    pcrlf
                  *
0174 8E   030C    pbddrn  ldx    #lbddrn    message output routines
0177 16   C1AD            lbra   exits
017A 8E   02E7    pbdsir  ldx    #lbdsir
017D 16   C1A7            lbra   exits
0180 8E   023C    pdform  ldx    #ldform
0183 16   C1A1            lbra   exits
0186 8E   0292    psrgtg  ldx    #lsrgtg
0189 16   C19B            lbra   exits
018C 8E   02BA    pnonst  ldx    #lnonst
018F 16   C195            lbra   exits
0192 8E   026A    pmemry  ldx    #lmemry
0195 16   C18F            lbra   exits
                  *
0198 20 20 20 63  lscrat  fcc    "-- copy entire diskette in drive "
01B9 30 20 74 6F  lscsdr  fcc    "0 to drive "

01C4 30 3F 20     lsctdr  fcc    "0? "
01C7 07 04                fcb    $07,$04
01C9 20 20 20 77  lwprot  fcc    "-- write-protect source"
01E0 20 64 69 73          fcc    " diskette now!"
01EE 07 0A 04             fcb    $07,$0a,$04
01F1 20 20 20 74  ltrgna  fcc    "-- target identification:"
020A 07 04                fcb    $07,$04
020C 20 20 20 73  lsrcna  fcc    "-- source identification:"
0225 07 04                fcb    $07,$04
0227 20 20 20 63  lcontn  fcc    "-- continue (y/n)? "
023A 07 04                fcb    $07,$04
023C 20 20 20 73  ldform  fcc    "-- source and target have"
0255 20 64 69 66          fcc    " different formats!"
0268 07 04                fcb    $07,$04
026A 20 20 20 6E  lmemry  fcc    "-- not enough memory for track"
```

```
0288 20 62 75 66          fcc    " buffer!"
0290 07 04                fcb    $07,$04
0292 20 20 20 73  lsrgtg  fcc    "-- source has more tracks than "
02B1 74 61 72 67          fcc    "target!"
02B8 07 04                fcb    $07,$04
02BA 20 20 20 73  lnonst  fcc    "-- source or target has"
02D1 20 6E 6F 6E          fcc    " nonstandard format!"
02E5 07 04                fcb    $07,$04
02E7 20 20 20 75  lbdsir  fcc    "-- unable to read system id "
0303 72 65 63 6F          fcc    "record!"
030A 07 04                fcb    $07,$04
030C 20 20 20 69  lbddrn  fcc    "-- invalid drive number!"
0324 07 04                fcb    $07,$04
                  *
C100                      org    cmdadr
                  *
C100 CC   0001    setup1  ldd    #$0001     set up for copy process
C103 FD   C5AE            std    srctks
C106 FD   C5B0            std    trgtks
C109 F7   C5BD            stb    lastrk
C10C B6   C5B7    xloop0  lda    trkfit
C10F B7   C5B6            sta    countr
C112 F6   C5BA            ldb    srcdrv
C115 17   01C9            lbsr   psourc     insert source disk
C118 17   0179            lbsr   drvsel     select source drive
C11B FC   C5AE            ldd    srctks
C11E 108E 0000            ldy    #buffer
C122 BE   C5B2            ldx    loctab
C125 8D   7F      xloop1  bsr    setimg     set up for read
C127 17   00CC            lbsr   doread     read sector
C12A 1029 01E7            lbvs   pferrd
C12E 4C                   inca
C12F B1   C5B8            cmpa   maxtrk     check max track - seek error
C132 22   05              bhi    xloop2
C134 7A   C5B6            dec    countr
C137 26   EC              bne    xloop1     loop back
C139 FD   C5AE    xloop2  std    srctks     save next track & sector
C13C B6   C5B7            lda    trkfit
C13F B7   C5B6            sta    countr
C142 F6   C5BB            ldb    trgdrv
C145 17   0183            lbsr   ptarge     insert target disk
C148 17   0149            lbsr   drvsel     select target drive
C14B FC   C5B0            ldd    trgtks
C14E 108E 0000            ldy    #buffer
C152 BE   C5B2            ldx    loctab
C155 8D   4F      xloop3  bsr    setimg     set up for write
C157 17   00CF            lbsr   dowrit     write it
C15A 1029 01BC            lbvs   pfermt
C15E 4C                   inca
C15F B1   C5B8            cmpa   maxtrk     check max track - seek error
C162 22   05              bhi    xloop4
C164 7A   C5B6            dec    countr
C167 26   EC              bne    xloop3     loop back
C169 FD   C5B0    xloop4  std    trgtks     save next track & sector
C16C B6   C5AE            lda    srctks
C16F B1   C5B8            cmpa   maxtrk
                  *
C174 F6   C5BB    restor  ldb    trgdrv     restore original sir
C177 17   0108            lbsr   getsir     read sir
C17A 1026 01A6            lbne   pbdvol
C17E 31   88 50           leay   $50,x      sir id info
C181 8E   C5BF            ldx    #idbuff    id hold area
C184 C6   0D              ldb    #$0d       13 bytes
C186 17   0113            lbsr   movexy     move it
C189 8E   C840            ldx    #sysfcb    write sir
C18C CC   0003            ldd    #$0003
C18F ED   88 1E           std    $1e,x
C192 86   0A              lda    #$0a
C194 A7   84              sta    ,x
C196 BD   D406            jsr    fms
C199 1026 0187            lbne   pbdvol
                  *
C19D 8E   C50C    pcompl  ldx    #lcompl    complete - exit
C1A0 BD   CD1E            jsr    pstrng
```

```
C1A3 7E   CD03           jmp   warms
                 *
                 * entry:
                 *   y=data address
                 *   x=beginning address of table
                 *   d=track/sector
                 * exit:
                 *   y=next data address
                 *   x=same as entry
                 *   b=sector+constant
                 *   a=track
C1A6 34   76     setimg  pshs  u,y,x,b,a
C1A8 33   E4             leau  ,s         set pointer
C1AA 32   7C             leas  -$04,s     make stack room
C1AC 31   A9 FF00        leay  $ff00,y    $ff00 is -256 (sector size)
C1B0 10AF 5C             sty   -$04,u     save address - $0100
C1B3 E7   5E             stb   -$02,u     save start sector
C1B5 108E C5AA           ldy   tabptr     point to table
C1B9 EC   A4             ldd   ,y
C1BB AB   44             adda  $04,u      get max track
C1BD A7   44             sta   $04,u
C1BF E7   5F             stb   -$01,u     save table constant
C1C1 6D   C4             tst   ,u         check for track zero
C1C3 27   02             beq   setim0
C1C5 31   22             leay  $02,y      adjust pointer
C1C7 10AE 22     setim0  ldy   $02,y      get table address
C1CA A6   5E     setim1  lda   -$02,u     get sector
C1CC A6   A6             lda   a,y        get physical sector
C1CE A7   80             sta   ,x+        store it
C1D0 5F                  clrb
C1D1 E3   5C             addd  -$04,u     find start value
C1D3 ED   81             std   ,x++       save start address
C1D5 A6   5E             lda   -$02,u     get sector
C1D7 4C                  inca             make next logical number
C1D8 A1   A4             cmpa  ,y         check for max sector
C1DA 23   02             bls   setim2
C1DC 86   01             lda   #$01       reset to first
C1DE A7   5E     setim2  sta   -$02,u     save sector
C1E0 A1   41             cmpa  $01,u      check for complete track
C1E2 26   E6             bne   setim1
C1E4 6F   84             clr   ,x         mark end - null
C1E6 A6   41             lda   $01,u      get start sector
C1E8 AB   5F             adda  -$01,u     add to constant
C1EA A1   A4             cmpa  ,y
C1EC 23   02             bls   setim3
C1EE A0   A4             suba  ,y
C1F0 A7   41     setim3  sta   $01,u      reset rb on null
C1F2 32   C4             leas  ,u         restore stack
C1F4 35   F6             puls  pc,u,y,x,b,a
                 *
C1F6 34   76     doread  pshs  u,y,x,b,a read a track
C1F8 33   E4             leau  ,s
C1FA 86   0C             lda   #$0c       max retries
C1FC 34   02             pshs  a
C1FE 1F   12             tfr   x,y
C200 20   02             bra   dorea2
C202 31   23     dorea1  leay  $03,y
C204 A6   C4     dorea2  lda   ,u         get track
C206 17   00AF           lbsr  displa     display track
C209 E6   A4             ldb   ,y         get sector
C20B 27   18             beq   dorea3
C20D AE   21             ldx   $01,y      get buffer address
C20F BD   DE00           jsr   dread      read the sector
C212 27   EE             beq   dorea1
C214 8E   C840           ldx   #sysfcb    error retry
C217 F6   C5BA           ldb   srcdrv
C21A E7   03             stb   $03,x
C21C BD   DE09           jsr   drest      restore
C21F 6A   5F             dec   -$01,u
C221 26   E1             bne   dorea2     retry
C223 1A   02             orcc  #$02       set error
C225 32   C4     dorea3  leas  ,u
C227 35   F6             puls  pc,u,y,x,b,a
                 *
```

```
C229 34   76     dowrit  pshs  u,y,x,b,a write a track
C22B 33   E4             leau  ,s
C22D 86   0C             lda   #$0c       max retries
C22F 34   02             pshs  a
C231 1F   12             tfr   x,y
C233 20   02             bra   dowri2
C235 31   23     dowri1  leay  $03,y
C237 A6   C4     dowri2  lda   ,u         get track
C239 17   007C           lbsr  displa     display track
C23C E6   A4             ldb   ,y         get sector
C23E 27   1C             beq   dowri5
C240 AE   21             ldx   $01,y      get buffer address
C242 BD   DE03           jsr   dwrite     write a sector
C245 27   EE             beq   dowri1
C247 8E   C840           ldx   #sysfcb    error retry
C24A F6   C5BB           ldb   trgdrv
C24D E7   03             stb   $03,x
C24F BD   DE09           jsr   drest      restore
C252 6A   5F             dec   -$01,u
C254 26   E1             bne   dowri2     retry
C256 1A   02     dowri3  orcc  #$02       force error
C258 32   C4     dowri4  leas  ,u
C25A 35   F6             puls  pc,u,y,x,b,a
C25C 10AE 42     dowri5  ldy   $02,u      verify target disk
C25F 20   02             bra   dowri7
C261 31   23     dowri6  leay  $03,y
C263 A6   C4     dowri7  lda   ,u         oet track
C265 E6   A4             ldb   ,y         get sector
C267 27   EF             beq   dowri4
C269 BD   DE18           jsr   dseek      seek track
C26C BD   DE06           jsr   dverfy     verify it
C26F 27   F0             beq   dowri6
C271 6A   5F     dowri8  dec   -$01,u
C273 27   E1             beq   dowri3     retry
C275 A6   C4             lda   ,u         get track
C277 E6   A4             ldb   ,y         get sector
C279 AE   21             ldx   $01,y      get buffer address
C27B BD   DE03           jsr   dwrite     rewrite the sector
C27E 27   E3             beq   dowri7
C280 20   EF             bra   dowri8
                 *
C282 8E   C840   getsir  ldx   #sysfcb    read sir
C285 E7   03             stb   $03,x
C287 CC   0003           ldd   #$0003     track 0 sector 3
C28A ED   88 1E          std   $1e,x
C28D 86   09             lda   #$09       single sector read
C28F A7   84             sta   ,x
C291 7E   D406           jmp   fms
                 *
C294 8E   C840   drvsel  ldx   #sysfcb    select drive
C297 E7   03             stb   $03,x
C299 7E   DE0C           jmp   drive
                 *
C29C A6   80     movexy  lda   ,x+        move b bytes from x to y
C29E A7   A0             sta   ,y+
C2A0 5A                  decb
C2A1 26   F9             bne   movexy
C2A3 39                  rts
                 *
C2A4 34   06     lookup  pshs  b,a        find max sector in table
C2A6 8E   C32D           ldx   #tabtop    point to table
C2A9 A6   80             lda   ,x+        get size
C2AB E1   84     lookul  cmpb  ,x         check sector count
C2AD 27   07             beq   looku2
C2AF 30   06             leax  $06,x
C2B1 4A                  deca
C2B2 26   F7             bne   lookul
C2B4 1C   FB             andcc #$fb       force error
C2B6 35   86     looku2  puls  pc,b,a
                 *
C2B8 B1   C5BD   displa  cmpa  lastrk     display track number
C2BB 27   12             beq   displx
C2BD 34   76             pshs  d,u,x,y
C2BF B7   C5BD           sta   lastrk
```

```
C2C2 8E    C58D             ldx    #lastrk   point to track
C2C5 8D    CD3C             jsr    outhex    display track
C2C8 86    0D               lda    #$0d      cr
C2CA 8D    CD0F             jsr    outch
C2CD 35    76               puls   d,u,x,y
C2CF 39            displx   rts
                   *
C2D0 8D    CD1E    putget   jsr    pstrng    display and wait for input
C2D3 8D    CD15             jsr    getchr
C2D6 81    03               cmpa   #$03      check for abort
C2D8 1027  0043             lbeq   pabort
C2DC 84    5F               anda   #$5f
C2DE 81    59               cmpa   #$59
C2E0 39                     rts
                   *
C2E1 7D    C58E    psourc   tst    single    insert source disk
C2E4 26    14               bne    psourx
C2E6 34    76               pshs   d,u,x,y
C2E8 8E    C521             ldx    #lsourc
C2EB BC    C5B4             cmpx   lasmsg
C2EE 27    05               beq    psourd
C2F0 BF    C5B4             stx    lasmsg
C2F3 8D    DB               bsr    putget
C2F5 BD    CD24    psourd   jsr    pcrlf
C2F8 35    76               puls   d,u,x,y
C2FA 39            psourx   rts
                   *
C2FB 7D    C58E    ptarge   tst    single    insert target disk
C2FE 26    14               bne    ptargx
C300 34    76               pshs   d,u,x,y
C302 8E    C548             ldx    #ltarge
C305 BC    C5B4             cmpx   lasmsg
C308 27    05               beq    ptargd
C30A BF    C5B4             stx    lasmsg
C30D 8D    C1               bsr    putget
C30F BD    CD24    ptargd   jsr    pcrlf
C312 35    76               puls   d,u,x,y
C314 39            ptargx   rts
                   *
C315 8E    C49A    pferrd   ldx    #lferrd   message output routines
C318 20    0D               bra    exits
C31A 8E    C4BF    pferwt   ldx    #lferwt
C31D 20    08               bra    exits
C31F 8E    C56F    pabort   ldx    #labort
C322 20    03               bra    exits
C324 8E    C4E4    pbdvol   ldx    #lbdvol
                   *
C327 BD    CD1E    exits    jsr    pstrng    print error and exit
C32A 7E    CD03             jmp    warms
                   *
C32D 0A            tabtop   fcb    $0a       number of entries in table
C32E 0A 08                  fcb    $0a,$08   sectors, format constant
C330 C442                   fdb    tab5ss    track = 0
C332 C442                   ldb    tab5ss    track > 0
C334 0F 03                  fcb    $0f,$03
C336 C36A                   fdb    tab8ss
C338 C36A                   fdb    tab8ss
C33A 12 02                  fcb    $12,$02
C33C C442                   fdb    tab5ss
C33E C44D                   fdb    tab5sd
C340 14 07                  fcb    $14,$07
C342 C460                   fdb    tab5ds
C344 C460                   fdb    tab5ds
C346 1A 04                  fcb    $1a,$04
C348 C36A                   fdb    tab8ss
C34A C399                   fdb    tab8sd
C34C 1D 12                  fcb    $1d,$12
C34E C36A                   fdb    tab8ss
C350 C3B4                   fdb    tab8se
C352 1E 03                  fcb    $1e,$03
C354 C37A                   fdb    tab8ds
C356 C37A                   fdb    tab8ds
C358 24 02                  fcb    $24,$02
C35A C468                   fdb    tab5ds
```

```
C35C C475                   fdb    tab5dd
C35E 34 04                  fcb    $34,$04
C360 C37A                   fdb    tab8ds
C362 C3D2                   fdb    tab8dd
C364 3A 20                  fcb    $3a,$20
C366 C37A                   fdb    tab8ds
C368 C407                   fdb    tab8de
                   *
                   * interlace tables
                   *  first element of each table is number
                   *    of sectors per track
                   *  remainder of each table is sectors in
                   *    interlace order
                   *  last three characters of table name:
                   *    size, sides, and density, respectively
                   *
C36A 0F 01 06 0B   tab8ss   fcb    $0f,$01,$06,$0b,$04,$09,$0e,$02
C372 07 0C 05 0A            fcb    $07,$0c,$05,$0a,$0f,$03,$08,$0d
C37A 1E 01 06 0B   tab8ds   fcb    $1e,$01,$06,$0b,$04,$09,$0e,$02
C382 07 0C 05 0A            fcb    $07,$0c,$05,$0a,$0f,$03,$08,$0d
C38A 17 1C 10 15            fcb    $17,$1c,$10,$15,$1a,$13,$18,$1d
C392 11 16 1B 14            fcb    $11,$16,$1b,$14,$19,$1e,$12
C399 1A 01 0E 09   tab8sd   fcb    $1a,$01,$0e,$09,$16,$04,$11,$0c
C3A1 19 07 14 02            fcb    $19,$07,$14,$02,$0f,$0a,$17,$05
C3A9 12 0D 1A 08            fcb    $12,$0d,$1a,$08,$15,$03,$10,$0b
C3B1 18 06 13               fcb    $18,$06,$13
C3B4 1D 01 06 0B   tab8se   fcb    $1d,$01,$06,$0b,$10,$15,$1a,$02
C3BC 07 0C 11 16            fcb    $07,$0c,$11,$16,$1b,$03,$08,$0d
C3C4 12 17 1C 04            fcb    $12,$17,$1c,$04,$09,$0e,$13,$18
C3CC 1D 05 0A 0F            fcb    $1d,$05,$0a,$0f,$14,$19
C3D2 34 01 0E 09   tab8dd   fcb    $34,$01,$0e,$09,$16,$04,$11,$0c
C3DA 19 07 14 02            fcb    $19,$07,$14,$02,$0f,$0a,$17,$05
C3E2 12 0D 1A 08            fcb    $12,$0d,$1a,$08,$15,$03,$10,$0b
C3EA 18 06 13 2D            fcb    $18,$06,$13,$2d,$1b,$28,$23,$30
C3F2 1E 2B 26 33            fcb    $1e,$2b,$26,$33,$21,$2e,$1c,$29
C3FA 24 31 1F 2C            fcb    $24,$31,$1f,$2c,$27,$34,$22,$2f
C402 1D 2A 25 32            fcb    $1d,$2a,$25,$32,$20
C407 3A 01 06 0B   tab8de   fcb    $3a,$01,$06,$0b,$10,$15,$1a,$02
C40F 07 0C 11 16            fcb    $07,$0c,$11,$16,$1b,$03,$08,$0d
C417 2F 34 39 21            fcb    $2f,$34,$39,$21,$26,$2b,$30,$35
C41F 3A 22 27 2C            fcb    $3a,$22,$27,$2c,$31,$36,$12,$17
C427 1C 04 09 0E            fcb    $1c,$04,$09,$0e,$13,$18,$1d,$05
C42F 0A 0F 14 19            fcb    $0a,$0f,$14,$19,$1e,$23,$28,$2d
C437 32 37 1F 24            fcb    $32,$37,$1f,$24,$29,$2e,$33,$38
C43F 20 25 2A               fcb    $20,$25,$2a
C442 0A 01 04 02   tab5ss   fcb    $0a,$01,$04,$02,$05,$03,$06,$09
C44A 07 0A 08               fcb    $07,$0a,$08
C44D 12 01 07 0D   tab5sd   fcb    $12,$01,$07,$0d,$02,$08,$0e,$03
C455 09 0F 04 0A            fcb    $09,$0f,$04,$0a,$10,$05,$0b,$11
C45D 06 0C 12               fcb    $06,$0c,$12
C460 14 01 04 02   tab5ds   fcb    $14,$01,$04,$02,$05,$03,$12,$10
C468 13 11 14 06            fcb    $13,$11,$14,$06,$09,$07,$0a,$08
C470 0B 0E 0C 0F            fcb    $0b,$0e,$0c,$0f,$0d
C475 24 01 07 0D   tab5dd   fcb    $24,$01,$07,$0d,$02,$08,$0e,$03
C47D 09 0F 16 1C            fcb    $09,$0f,$16,$1c,$22,$17,$1d,$23
C485 18 1E 24 04            fcb    $18,$1e,$24,$04,$0a,$10,$05,$0b
C48D 11 06 0C 12            fcb    $11,$06,$0c,$12,$19,$1f,$13,$1a
C495 20 14 1B 21            fcb    $20,$14,$1b,$21,$15
                   *
C49A 20 20 20 66   lferrd   fcc    "-- fatal error reading source disk!"
C4BD 07 04                  fcb    $07,$04
C4BF 20 20 20 66   lferwt   fcc    "-- fatal error writing target disk!"
C4E2 07 04                  fcb    $07,$04
C4E4 20 20 20 75   lbdvol   fcc    "-- unable to restore system id "
C503 72 65 63 6F            fcc    "record!"
C50A 07 04                  fcb    $07,$04
C50C 20 20 20 6D   lcompl   fcc    "-- mirror complete!"
C51F 07 04                  fcb    $07,$04
C521 20 20 20 69   lsourc   fcc    "-- insert source disk and hit a key. "
C546 07 04                  fcb    $07,$04
C548 20 20 20 69   ltarge   fcc    "-- insert target disk and hit a key. "
C56D 07 04                  fcb    $07,$04
C56F 20 20 20 6D   labort   fcc    "-- mirror aborted!"
C581 0D 0A                  fcb    $0d,$0a
```

```
C583 20 20 20 74          fcc    "-- target disk should be reformatted!"
C5A8 07 04                fcb    $07,$04
                   *
C5AA               tabptr  rmb    4         local variables
C5AE               srctks  rmb    2
C5B0               trgtks  rmb    2
C5B2               loctab  rmb    2
C5B4               lasmsg  rmb    2
C5B6               countr  rmb    1
C5B7               trkfit  rmb    1
C5B8               maxtrk  rmb    1
C5B9               maxsec  rmb    1
C5BA               srcdrv  rmb    1
C5BB               trgdrv  rmb    1
C5BC               verflg  rmb    1
C5BD               lastrk  rmb    1
C5BE               single  rmb    1
C5BF               idbuff  rmb    13        sir bold area
                   *
              0000 buffer  equ    start     beginning addr of buffer
                   *
                          end    start

0 ERROR(S) DETECTED
```

```
*
*  flex system defined entry points and equates
*
flex       equ      $c000
*
linbuf     equ      $c080                line buffer
cmdadr     equ      $c100                utility command space (1.5 k)
cmdend     equ      $c700                utility command space end
sysfcb     equ      $c840                system fcb address
*
*  global values specified by ttyset and asn
*
bspchr     equ      $cc00                backspace character
delchr     equ      $cc01                delete character
eolchr     equ      $cc02                end of line character
depth      equ      $cc03                depth count
width      equ      $cc04                width count
nulls      equ      $cc05                null count
tabchr     equ      $cc06                tab character
bsechr     equ      $cc07                backspace echo character
pause      equ      $cc09                pause control byte
escchr     equ      $cc0a                escape character
s_drn      equ      $cc0b                system drive number
w_drn      equ      $cc0c                working drive number
*
*  flex system global variables
*

sysflg     equ      $cc0d                use system drive flag
sysdate    equ      $cc0e                date registers
lsttrm     equ      $cc11                last terminator character
cbufpt     equ      $cc14                line buffer pointer
escret     equ      $cc16                escape return register
curchr     equ      $cc18                current nxtch character
prevch     equ      $ccf9                previous nxtch character
curlct     equ      $cc1a                current line count
loadao     equ      $cc1b                loader address offset data
xfrflg     equ      $ccfd                transfer address flag
xfradr     equ      $cc1e                transfer address of loaded file
outswt     equ      $cc22                output switch
inswt      equ      $cc23                input switch
docmdf     equ      $cc28                docmd entry flag
curcol     equ      $cc29                current output column
memend     equ      $cc2b                end of memory address
cputype    equ      $cc33                cpu type flag
retadr     equ      $cc43                docmd return address
ulcflag    equ      $cc49                upper/lower case map flag
prompt     equ      $cc4e                pointer to prompt string
*
*  cpu type flag bit definitions
```

```
*
cpu_2mhz   equ     %10000000         1 => 2 mhz cpu clock rate
cpu_slow   equ     %01000000         1 => memory stretch active
cpu_50hz   equ     %00100000         1 => 50 hz power line frequency
cpu_ramf   equ     %00010000         1 => cpu ram is available
cpu_rtck   equ     %00001000         1 => 6819 real time clock avail
cpu_iobx   equ     %00000100         1 => i/o set up like old box
cpu_time   equ     %00000010         1 => 6840 timer available
cpu_xmem   equ     %00000001         1 => extended memory is used
*
*  printer driver interface addresses
*
pt_rap     equ     $cc35             printer reserved area pointer
pt_len     equ     $cc37             printer reserved area length
pt_dev     equ     $cc39             printer device address
pinit      equ     $ccc0             printer initialize vector
pterm      equ     $ccd0             printer close vector
pcbk       equ     $ccd8             printer ready check vector
pout       equ     $cce4             printer character output module
prcflg     equ     $ccfc             active spooling process flag
*
*  flex system defined entry vectors
*
colds      equ     $cd00             flex cold start address
warms      equ     $cd03             flex warm start address
renter     equ     $cd06             re-enter flex processing
inch       equ     $cd09             input character (low level)
outch      equ     $cd0f             output character (low level)
getchr     equ     $cd15             input character routine
putchr     equ     $cd18             output character routine
inbuff     equ     $cd1b             input line buffer
pstrng     equ     $cd1e             print string
class      equ     $cd21             classify character
pcrlf      equ     $cd24             print cr/lf sequence
nxtch      equ     $cd27             get next character from input
getfil     equ     $cd2d             scan file spec address

load       equ     $cd30             load file entry point
setext     equ     $cd33             set up file extension
outdec     equ     $cd39             output decimal number
outhex     equ     $cd3c             output hexadecimal number
rpterr     equ     $cd3f             i/o error abort routine
gethex     equ     $cd42             get hexidecimal specification
outadr     equ     $cd45             output hexadecimal address
indec      equ     $cd48             get decimal number
docmd      equ     $cd4b             docmd entry address
status     equ     $cd4e             check terminal input status
*
*  low level terminal and interrupt control addresses
*
intap      equ     $d3de             vector for input tap routine
dummy      equ     $d3e0             dummy rts instruction for re
setirq     equ     $d3e1             set irq process vector
clrirq     equ     $d3e3             clear irq process vector
linch      equ     $d3e5             low-level term input w/o echo
t_off      equ     $d3ed             timer off routine address
t_on       equ     $d3ef             timer on routine address
t_init     equ     $d3f1             timer initialize routine addr
tinit      equ     $d3f5             low-level terminal initialize
tcheck     equ     $d3f7             low-level terminal check addr
toutch     equ     $d3f9             low-level terminal output addr
tinche     equ     $d3fb             low-level terminal input & echo
*
*  file management system entry points
*
fmscls     equ     $d403             close up all files entry
fms        equ     $d406             file manager exec call
fcbase     equ     $d409             file control block base
verify     equ     $d435             fms verify flag
surtab     equ     $d436             fms surname table
*
fcblen     equ     256+64            file control block length
*
*  disk driver entry points
*
```

```
dread    equ    $de00          read sector routine
dwrite   equ    $de03          write sector routine
dverfy   equ    $de06          verify routine
drest    equ    $de09          drive restore routine
drive    equ    $de0c          drive select routine
dcheck   equ    $de0f          check drive ready
dquick   equ    $de12          quick check drive ready
dseek    equ    $de1b          drive seek-to-sector routine
*
```

BUILD YOUR OWN FAT MAC

by

Mike Wolf
3195 Arizona
Los Alamos, NM 87544

I guess the big news is that I'm writing this on my FAT Mac. I just upgraded to 512K. There were no unexpected problems and the software I have tried so far seems to work OK and takes advantage of the added RAM. I can dimension a array of 45,000 double precision numbers in Microsoft Basic. Or, I can type a article of 100 pages or so in MacWrite (this is a calculated number. I haven't had time to write that much). More good news is that the RAMs have just taken a plunge in price, so the cost isn't completely out of reason.

Now for the bad news. This is not a job for the novice hardware hacker as it requires unsoldering 16 ICs from a **4-Layer** PCB. It also, of course, negates any warranty, and probably will make it difficult to get service through normal channels. However, for those of you who know no fear, here's how to do it.

This procedure is for the **Original Board**. It is my understanding that Apple has a new board out for the Fat Mac which will be installed in the 128K Macs when present stocks are depleted. I have not seen one, but I understand that you will only have to replace the RAMs in the new board.

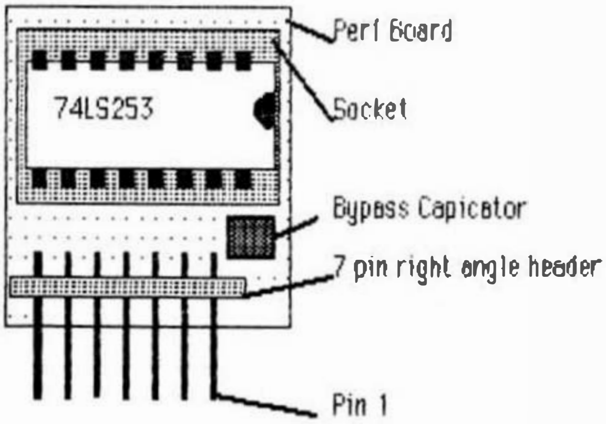
a) You will need the following tools: A*5 torklok screwdriver long enough to reach the screws under the handle. A conventional screwdriver can be modified for this task by grinding the blade to fit in the splines of the screws. A temperature controlled soldering iron with a fine tip suitable for high density printed circuit work. A good quality desoldering tool such as a Solduvac. Don't scrimp on the desoldering setup! A ruined board is a pretty expensive way to learn.

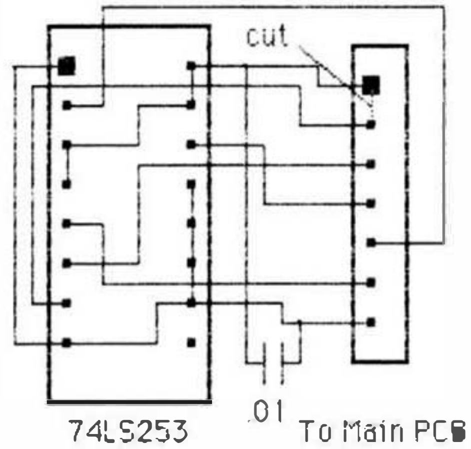
b) You will need the following parts: 16 256K Dynamic RAMS (41256 type), a small piece of perf board large enough to hold 1 16 pin IC, a 7 pin strip of right angle header (more about this later), a bypass capicator (.01 to .1 micro farad caramic), 1 74LS253 IC, and 17 Top quality IC sockets. Don't scrimp on these sockets. Use the machined pin type that cost about $1 apiece. I have used both Hitachi and NEC RAMs for this conversion; the Hitachi part number is HM50256-20, the NEC number is D41256-20. I see no reason that other parts of this type shouldn't work also. 200nS parts are fast enough.

c) Remove the five screws holding the case together (2 under the handle, 2 at the bottom of the back at the ends of the connectors, and one under the battery cover). Place the computer face down on a padded surface and remove the rear cover. It fits tightly and will need some persuading. Careful prying in the crack, jerking the rear cover, etc., to separate it. Once the rear cover is off you will see the board which fits parallel to the bottom of the case; this is the Computer Board. Remove it by disconnecting the two connectors (one to the disc, the other to the vertical board) and slide the board to the rear.

d) Set the computer aside and examine the board. Along the front center of the board is the RAM chips in 2 rows of 8 ICs each. On the left end of the CPU is a row of 7 solder filled holes. Once you have identified these items, get out your desoldering equipment and go to work. Clean the solder out of the 7 holes, and desolder and remove the 16 RAM chips. Be careful!! Clean the board carefully to remove flecks of solder that might cause shorts, and examine it (preferrabilly with a magnifier) to insure that no traces have been broken. If they have, repair them with stripped wire-wrap wire. Then Install 16 sockets in the RAM locations, and install the new RAM chips in the sockets.

e) Construct the multiplexer as shown in the illustrations, wiring it as shown. Cut the trace beween the square padded hole (pin 1) and the second hole of the row of holes you just cleaned, out and install the multiplexer in place. The board should sit vertical to the computer board. Check your work twice: are the ICs firmly in the sockets with no bent pins? Is Pin *1 to the front of the board? Clean the flux off with a solvent and old toothbrush. Check for shorts. When you are satisfied, reinsert the board into the case and reconnect the cables. Plug it in and turn it on. If you get the usual beep and sign-on, followed by the no-disc icon, all is well. Insert a boot disc and check it out. If you have MicroSoft BASIC, run it and try "**?fre(0)**" you should get 374000 or so. Run some other stuff to check it out, and then close it up. You've just saved enough to buy a second disc drive!!





FLEX 2/9 CATALOG UTILITY

FLEX 2/9 CATALOG UTILITY (CAT.CMD)

DONALD E. GOULETTE
BOX 1153
FABENS, TEXAS 79838
PHONE 915-764-3607

This article describes an enhancement of the old TSC FLEX 2.0 CAT.CMD utility. This new version should be usable for both FLEX 2 and 9 systems. I can only vouch for my FLEX 2 configuration. I bought my SW6800 back in the days before the Earth cooled, and have put several thousand hours into software development for it. The following listing is the result of finally sitting down and doing something about the limitations of the old CAT.CMD.

CAT.CMD is pretty much compatible with the syntax of the old CAT.CMD utility. The main difference being that the input line parameters are ALL tested for syntax first; redundant options and match lists are ignored; and drive numbers, options and match lists can be entered in any order.

New features are listing of catalog protected files, deleted files, and two forms of printout.

Listing of catalog protected files ('P' option) is assessed by using a 3 character password (optional) for those files that you have misplaced.

Listing of deleted files ('D' option) is handy when you have inadvertently (like 2:00 in the A.M.) deleted a file by mistake (but not over written) and you use some sort of disk recovery utility in conjunction with CAT.CMD, such as the DISKSAVE.CMD utility.

The short form of printout (default) is ideal to view all the file names on one page, when your only concern is the presence of the names and there extensions. The listing can be as many as 9 names wide ('C' option) or as few as 1. A special 'CO' option will list in the old FLEX CAT.CMD format.

The expanded form ('E' option) will list the file number, drive number the file was found on, file name, file extension, file protection (if any), file start track-sector, and file end track-sector (again used in conjunction with a disk recovery utility).

The assembled listing is pretty much self documenting and should be fairly easy to make any changes as far as password, option codes, and default conditions.

CONCLUSION

Believe it or not, this utility started out as an experiment with the FLEX documented subroutines and just snowballed ,after a month, into its present form. It sure has been an experience and I enjoyed it inmeasurably.

I've been a subscriber to '68 since day one, and I'm glad to finally contribute something.

- - -

```
        NAM    CAT.CMD

*        CAT.CMD UTILITY  (ENHANCED VERSION)

*        DONALD GOULETTE
*        BOX 1153
*        FABENS, TEXAS 79838
*        (915)-764-3607

*        26 MARCH 1984


*SYNTAX FOR 'CAT.CMD':

*1.      +++CAT
*2.      +++CAT +E 1 0 +D
*3.      +++CAT +EPDOG 0 .SYS P.CMD A
*4.      +++CAT 0,.TXT,+C5,1,+PDOG
*5.      +++CAT 0 +C0
*6.      +++ AT .CMD +DC5PDOGE 0 1 .CMD COPY.CMD

*LINE 1: CAT WORKING DRIVE (ALL FILES) OR BOTH DRIVES
*        IF WORKING DRIVE = AUTO. (LISTED IN SHORT FORM)

*LINE 2:  AT, IN EXPANDED FORM, DRIVES 0 AND 1 AND THERE
*        DELETED FILES.

*LINE 3: CAT, EXPANDED FORM, DRIVE 0, ALL FILES WITH
*        .SYS EXTENSION (ALSO CAT PROTECTED, IF ANY),
*        ANY FILE THAT STARTS WITH 'P' AND ALSO A '.CMD'
*        EXTENSION (ALSO CAT PROTECTED, IF ANY), AND
*        ANY FILE STARTING WITH AN 'A' (ALSO CAT PRO-
*        TE TED, IF ANY).

*LINE 4: CAT,DRIVE 0 AND 1, ALL '.TXT' FILES (ALSO
*         AT PROTECTED, IF ANY). IN SHORT FORMAT, 5
*        COLUMNS WIDE.
*        SPACES AND COMMAS ARE INTERCHANGABLE.

*LINE 5: CAT DRIVE 0 (ALL FILES) IN THE OLD CAT.CMD
*        FORMAT.

*LINE 6: CAT, DRIVES 0 AND 1, ALL FILES WITH A '.CMD'
*        EXTENSION AND ALSO ALL '.CMD' CAT PROTECTED
*        AND DELETED FILES, IF ANY. (EXPANDED FORM)
*        NOTE: THE 2ND .CMD INSTRUCTION IN THE LINE
*        AND THE COPY.CMD ARE REDUNDANT AND ARE IGNORED.
*        ALSO, THE 'E' IN THE OPTION LINE INHIBITS THE
*        'C5' OPTION REQUEST (SHORT FORM ONLY).


*EXPLANATION OF OPTIONS:

*PREFIX ALL OPTIONS WITH "OPTCHR", IN THIS CASE "+".
*OPTION MAY BE ENTERED INDIVIDUALLY OR TOGETHER WITH OTHERS.

*EX: +DEPDOG or +D +E +PDOG


*OPTION "D":  LIST A DELETED FILE IF ALL OTHER DRIVE, MATCH
*             CONDITIONS HAVE BEEN MET.

*EX:   +++CAT +D .CMD
*(DELETED FILE XYZ.TXT WOULD NOT BE LISTED BUT DELETED FILE
*XYZ.CMD WOULD BE).

*NOTE: A DELETED FILE WILL HAVE A '?' INSERTED AT THE START
*OF THE FILE NAME (THE FIRST LETTER OF THE NAME OF A DELETED
*FILE IS UNKNOWN BECAUSE FLEX REPLACES IT WITH A $FF).

*OPTION "P":  LIST A CAT PROTECTED FILE IF ALL OTHER DRIVE,
*             MATCH CONDITIONS HAVE BEEN MET.

*EX:   +++CAT +PDOG .SYS
*(CAT PROTECTED FILE ABC.CMD WOULD NOT BE LISTED, BUT
*CAT PROTECTED FILE ABC.SYS WOULD BE).

*NOTE: FOLLOWING THE 'P' OPTION, A 3 ALPHA-NUMERIC CODE MUST
*      BE INSERTED. IF AN INCORRECT 3 PART CODE OR NONE, A
```

```
                  *        SYNTAX ERROR WILL BE ISSUED.
                  *        CHANGE THE STRING AT LABEL 'KEY' FOR ANOTHER PASSWORD.
                  *        CHANGE THE EQU AT LABEL 'PASS' TO 'N' TO DEFEAT THE
                  *        PASSWORD FEATURE.


                  *OPTION "E":  CAT IN THE EXPANDED FORM (IF NOT USED, DEFAULT
                  *                TO THE SHORT FORM).

                  *EX:   +++CAT .CMD +E
                  *(PRINT ALL WORKING DRIVE .CMD FILES IN EXPANDED FORM).

                  *EXPANDED FORM PRINTS THE FOLLOWING:
                  *FILE NUMBER, DRIVE # OF FILE, FILE NAME, FILE EXTENSION,
                  *COPY DATE, FILE PROTECTION IF ANY, FILE START TRACK-SECTOR.
                  *FILE END TRACK-SECTOR.....
                  *
                  *SHORT FORM IS A SHORT VERSION OF THE EXPANDED FORM.


                  *OPTION "C":  SET THE COLUMN WIDTH FOR SHORT FORM WIDTH OF
                  *             DISPLAY. (THIS OPTION IS DEFEATED IF 'E' OPTION
                  *             IS ENABLED).

                  *        CHANGE THE EQU AT LABEL 'CLMDFT' FOR A DIFFERENT
                  *        DEFAULT COLUMN COUNT.

                  *NOTE:   FOLLOWING THE 'C' OPTION, A NUMBER FROM 0 TO 9 MUST
                  *        BE USED OR A SYN AX ERROR WILL BE ISSUED.
                  *        'C0' IS A SPECIAL CASE, THAT WILL ENABLE THE OLD
                  *        CAT.CMD FORMAT.


                  *DOS EQUATES

                  *WITH THE FOLLOWING EQUATE, USE $A000 (FLEX 2.0)
                  *OR $C000 (FLEX 9.0).
                  *NOTE: FLEX 9.0 SHOULD WORK BUT I DON'T REALLY KNOW
                  *BECAUSE I'M ONLY RUNNING FLEX 2.0

C000              FLEX    EQU     $C000       START OF FLEX 6809

CC0C              WRKDRV  EQU     FLEX+$C0C WORK DRIVE #
CC11              LSTTRM  EQU     FLEX+$C11 LAST NON-ALPHA CHR
CC14              LBPNTR  EQU     FLEX+$C14 LINE BUFFER PNTR
CD03              WARMS   EQU     FLEX+$D03 DOS WARM START ENTRY
CD18              PUTCHR  EQU     FLEX+$D18 PUT CHR ROUTINE
CD1E              PMSG    EQU     FLEX+$D1E CRLF+PRNT STRING
CD24              PCRLF   EQU     FLEX+$D24 PRINT CR/LF
CD27              NXTCH   EQU     FLEX+$D27 GET NXT BUF CHR
CD39              OUTDEC  EQU     FLEX+$D39 OUTPUT A DEC #
CD3F              RPTERR  EQU     FLEX+$D3F REPORT DISK ERROR
CD45              OUTADR  EQU     FLEX+$D45 OUTPUT HEX ADDR

                  *FMS EQUATES

D403              FMSCLS  EQU     FLEX+$1403
D406              FMS     EQU     FLEX+$1406

                  *SYSTEM EQUATES

C840              FCB     EQU     FLEX+$840 FILE CONTROL BLOCK


                  *CHANGE THE FOLLOWING EQUATE TO ANY OF THESE:   !#$%&*+-

002B              OPTCHR  EQU     '+          OPTION LEAD IN CHR

                  *CHANGE THE FOLLOWING 4 EQUATES TO ANY OF THESE: A to Z

0045              EXPCHR  EQU     'E          EXPANDED MODE
0044              DELCHR  EQU     'D          PRINT DELETED FILES
0050              PROCHR  EQU     'P          PRINT CAT PROT FILES
0043              CLMCHR  EQU     'C          SHORT FORM CLM CNT


                  *IF THE PASSWORD MECHANISM IS NOT WANTED THEN MAKE
                  *THE FOLLOWING EQUATE AN 'N':

0059              PASS    EQU     'Y          PASSWORD MECH FLAG

                  *THE FOLLOWING IS THE 3 PART PASSWARD. USE ONLY ALPHA
                  *NUMERICAL CHARACTERS. EX: CAT XYZ 007 GH2

0044              KEY     EQU     'D
004F              KEY1    EQU     'O
0047              KEY2    EQU     'G

                  *IF YOU SO DESIRE, YOU MAY
                  *CHANGE THE FOLLOWING SHORT FORM DEFAULT CLM CNT: 0 to 9
                  *IF ZERO (0) IS USED THEN THE OLD STYLE CAT.CMD
                  *FORMAT WILL ENABLED......

0000              CLMDFT  EQU     0           SHORT FORM DEFAULT CLM CNT




                              *CAT.CMD UTILITY STARTS HERE

C100                                  ORG     FLEX+$100 UTILITY AREA

C100 20    3F             CAT         BRA     CAT0          GET AROUND TEMPS
C102 02                   VN          FCB     2             VERSION NUMBER
C103                      STORE       RMB     2             STORAGE
C105                      SIZE        RMB     2             "
C107                      PYRS        RMB     2             STRING PNTR SOURCE
```

```
C109                  PTRD    RMB     2         STRING PNTR DESTINATION
C10B                  CLMCNT  RMB     1         CLM CNTR
C10C                  EXPFLG  RMB     1         EXPAND FORM FLAG
C10D                  MATFLG  RMB     1         MATCH FLAG
C10E                  AUTFG   RMB     1         AUTO FLAG
C10F                  DEFFG   RMB     1         DEFAULT FLAG
C110                  FILCNT  RMB     2         FILE CNT
C112                  DRVFG   RMB     1         DRV ENABLE FLAGS
C113                  DELFG   RMB     1         CAT DEL FILE FLAG
C114                  CATFG   RMB     1         CAT PROT FLAG
C115                  CLMLIM  RMB     1         MAX SHORT FORM CLM CNT
C116                  OLDFG   RMB     1         OLD CAT FORMAT
C117                  IOXTMP  RMB     2         ADDR OF PNTR
C119                  STGPTR  RMB     40        MATCH PNTRS STACK

C141 7F    C10C       CAT0    CLR     EXPFLG    SHORT FORM MODE
C144 7F    C10B               CLR     CLMCNT    INIT CLM CNTR
C147 7F    C10D               CLR     MATFLG    STRING MATCH FLAG
C14A 7F    C10F               CLR     DEFFG     DEFAULT FLAG
C14D 7F    C113               CLR     DELFG     CAT DEL FILE FLAG
C150 7F    C112               CLR     DRVFG     CLR ALL DRVS
C153 7F    C114               CLR     CATFG     CAT PROT FLAG
C156 7F    C10E               CLR     AUTFG     AUTO FLAG
C159 7F    C110               CLR     FILCNT    FILE COUNTER
C15C 7F    C111               CLR     FILCNT+1
C15F 7F    C116               CLR     OLDFG     OLD CAT FORM OFF
C162 86    FF                 LDAA    #$FF      MATCH STRING EOT
C164 B7    C119               STAA    STGPTR    INIT IT
C167 4F                       CLRA              ZERO
C168 81    00                 CMPA    #CLMDFT   ZERO MODE?
C16A 26    07                 BNE     OLD1      NOT OLD FORM
C16C 7C    C116               INC     OLDFG     OLD CAT FORM ON
C16F 86    01                 LDAA    #1        1 CLM W/ OLD
C171 20    02                 BRA     OLD1A
C173 86    00         OLD1    LDAA    #CLMDFT   DEFAULT CLM CNT
C175 B7    C115       OLD1A   STAA    CLMLIM    SAVE IT
C178 B6    CC0C               LDAA    WRKDRV    FETCH DEF DRV
C17B 81    FF                 CMPA    #$FF      AUTO MODE?
C17D 26    05                 BNE     NAMDE     NOT AUTO MODE
C17F 86    02                 LDAA    #2        DO BOTH DRVS
C181 B7    C10E               STAA    AUTFG

C184 B6    CC11       NAMDE   LDAA    LSTTRM    LAST CHR
C187 81    0D                 CMPA    #$D       CR?
C189 26    14                 BNE     CAT01     NO
C18B 8E    C840               LDX     #FCB      PNTR
C18E B6    CC0C               LDAA    WRKDRV    DEF WRK DRV
C191 A7    03                 STAA    3,X       SET DEF DRV
C193 81    FF                 CMPA    #$FF      AUTO MODE?
C195 26    05                 BNE     CAT2V     NO
C197 86    02                 LDAA    #2        CAT BOTH DRVS
C199 B7    C10E               STAA    AUTFG
C19C 7E    C22D       CAT2V   JMP     CAT2      NO OPTIONS, GO DO IT
C19F BD    CD27       CAT01   JSR     NXTCH     GET DRV
C1A2 24    20                 BCC     CAT02     ALPHA NUMERIC
C1A4 81    0D         CAT01B  CMPA    #$0D      ALL DONE?
C1A6 1027  0083               LBEQ    CAT2      ALL DONE SCAN
C1AA BD    C34A               JSR     OPTION    TEST FOR OPTIONS
C1AD 25    12                 BCS     ERRYX     SYNTAX ERROR
C1AF 81    0D                 CMPA    #$D       CR?
C1B1 27    7A                 BEQ     CAT2      ALL DONE SCAN
C1B3 81    2C         CAT01A  CMPA    #',       COMMA?
C1B5 27    E8                 BEQ     CAT01     OK, SKIP
C1B7 81    20                 CMPA    #$20      SPACE?
C1B9 27    E4                 BEQ     CAT01     OK, SKIP
C1BB 81    2E                 CMPA    #'.       EXT?
C1BD 26    02                 BNE     ERRYX     SYNTAX ERROR
C1BF 20    1E                 BRA     MAT0      SET FLAG
C1C1 7E    C542       ERRYX   JMP     CAT6B     SYNTAX ERROR
C1C4 81    34         CAT02   CMPA    #'4       TOO LARGE
C1C6 24    17                 BCC     MAT0      SET MATCH STRING
C1C8 84    03                 ANDA    #$03      STRIP ASCII
C1CA 4C                       INCA              OFFSET
C1CB 5F                       CLRB
C1CC 1A    01                 SEC               SET DRV BIT
C1CE 59               ROT     ROLB              ROLL IN NEW DRV
C1CF 1C    FE                 CLC
C1D1 4A                       DECA
C1D2 26    FA                 BNE     ROT       ROTATE MORE
C1D4 FA    C112               ORAB    DRVFG     MASK IN DRV #
C1D7 F7    C112               STAB    DRVFG     UPDATE
C1DA 7C    C10F               INC     DEFFG     NO DEF DRV
C1DD 20    C0                 BRA     CAT01     KEEP SCANNING
C1DF BE    CC14       MAT0    LDX     LBPNTR    LINE BUFFER PNTR
C1E2 30    1F                 DEX               PRESENT CHR
>C1E4 BD   C1FA               JSR     SAVPTR    SAVE START OF STRING ADR
C1E7 24    03                 BCC     MAT1      NO ERR
C1E9 7E    C542               JMP     CAT6B     SYNTAX ERR
C1EC 7C    C10D       MAT1    INC     MATFLG    MATCH FLAG ON
C1EF BD    CD27       MAT1A   JSR     NXTCH     FIND END OF STRING
C1F2 24    FB                 BCC     MAT1A     AND LOCKOUT ANY STRING #
C1F4 81    2E                 CMPA    #'.       PERIOD (EXT) ?
C1F6 27    F7                 BEQ     MAT1A     SKIP EXT
C1F8 20    AA                 BRA     CAT01B    CONTINUE

                      *SEARCH FOR PLACE ON STRING STACK TO PUT NEW
                      *MATCH STRING POINTER...

C1FA BF    C117       SAVPTR  STX     IOXTMP    SAVE NEW PNTR
C1FD 8E    C119               LDX     #STGPTR   STRING VECTORS
C200 A6    84         SAV1    LDAA    0,X       FETCH
C202 81    FF                 CMPA    #$FF      END OF VECTORS?
C204 27    0C                 BEQ     SAV2      YES
C206 30    01                 INX               SEARCH FOR END
C208 30    01                 INX
C20A 8C    C141               CPX     #CAT0     TOO FAR?
C20D 26    F1                 BNE     SAV1      NOT YET!
C20F 1A    01                 SEC               ERR, OVERFLOW
```

```
C211 39                 RTS
C212 B6    C117   SAV2  LDAA   IDXTMP      FETCH
C215 A7    84           STAA   0,X         SAVE MSB
C217 30    01           INX
C219 B6    C118         LDAA   IDXTMP+1    FETCH
C21C A7    84           STAA   0,X         SAVE LSB
C21E 86    FF           LDAA   #$FF        EOT
C220 30    01           INX                SET NEW EOT
C222 A7    84           STAA   0,X         SET
C224 BE    C119         LDX    #STG TR     1ST ADDR OF MATCH VECTOR
C227 BF    C117         STX    IDXTMP      START OF VECTORS
C22A 1C    FE           CLC                OK
C22C 39                 RTS

                  *AT THIS POINT, THE INPUT LINE HAS BEEN TESTED...

C22D BD    CD24   CAT2  JSR    PCRLF
C230 7F    C108         CLR    CLMCNT      RESET
C233 7F    C111         CLR    FILCNT+1    FILE CNTR
C236 7D    C112         TST    DRVFG       ANY DRVS LEFT?
C239 27    2F           BEQ    CAT2Z       SKIP, DONE
C23B 5F                 CLRB
C23C B6    C112         LDAA   DRVFG
C23F 84    01           ANDA   #1          DRV 0?
C241 26    18           BNE    FND0        YES
C243 B6    C112         LDAA   DRVFG
C246 84    02           ANDA   #2          DRV 1?
C248 2     13           BNE    FND1        YES
C24A B6    C112         LDAA   DRVFG
C24D 84    04           ANDA   #4          DRV 2?
C24F 26    0B           BNE    FND2        YES
C251 B6    C112         LDAA   DRVFG
C254 84    08           ANDA   #8          DRV 3?
C256 26    03           BNE    FND3        YES
C258 7E    C542         JMP    CAT6B       SYNTAX ERROR
C25B 5C           FND3  INCB               SET TO CURRENT DRV #
C25C 5C           FND2  INCB
C25D 5C           FND1  INCB
C25E 43           FND0  COMA               INV MASK
C25F B4    C112         ANDA   DRVFG
C262 B7    C112         STAA   DRVFG       UPDATE
C265 BE    C840         LDX    #FCB
C268 E7    03           STAB   3,X         SET TO CURRENT DRV #
C26A 7D    C10F   CAT2Z TST    DEFFG       DEFAULT DRV?
C26D 26    1C           BNE    CAT2A       NO
C 6F BE    C840         LDX    #FCB
C272 B6    CC0C         LDAA   WRKDRV      DEFAULT DRV
C275 A7    03           STAA   3,X         SET DRV
C277 7D    C10E         TST    AUTFG       AUTO SEARCH?
C27A 27    0F           BEQ    CAT2A       NO
C27C B6    C10E         LDAA   AUTFG       DRV 0 OR 1
C27F 81    02           CMPA   #2          DRV 0 FIRST?
C281 26    04           BNE    CAT2X       NO, DRV 1
C283 86    00           LDAA   #0          DRV 0
C285 20    02           BRA    CAT2Y       SKIP
C287 86    01     CAT2X LDAA   #1          DRV 1
C289 A7    03     CAT2Y STAA   3,X         SET DRV
C28B BE    C840   CAT2A LDX    #FCB
C28E 86    10           LDAA   #$10        OPEN SYS INFO
C2   A7    84           STAA   0,X         OPEN
C292 BD    0406         JSR    FMS
C 95 27    06           BEQ    CAT2B
C297 7E    C542   ERRY  JMP    CAT6B       SYNTAX ERROR
C29A 7E    C53C   ERRZ  JMP    CAT9        FMS DISK ERROR
C29D BE    C840   CAT2B LDX    #FCB
C2A0 86    07           LDAA   #$7         GET SYS INFO
C2A2 A7    84           STAA   0,X
C2A4 BD    0406         JSR    FMS
C2A7 26    F1           BNE    ERRZ        FMS ERROR
C2A9 BE    C605         LDX    #STG1       HEADER
C2AC BD    CD1E         JSR    PMSG
C2AF BE    C842         LDX    #FCB+2
C2B2 6F    84           CLR    0,X
C2B4 5F                 CLRB               NO SPACES
C2B5 BD    CD39         JSR    OUTDEC      PRINT DRV#
C2B8 BE    C655         LDX    #STG3       DISK#
C2BB BD    CD1E         JSR    PMSG
C2BE BE    C844         LDX    #FCB+4      SYS NAME
C2C1 C6    08           LDAB   #8          8 CHRS
C2C3 A6    84     AGNH  LDAA   0,X         FETCH CHR
C2C5 27    08           BEQ    ADGK        DONE
C2C7 BD    CD18         JSR    PUTCHR
C2CA 30    01           INX
C2CC 5A                 DECB
C2CD 26    F4           BNE    AGNH        MORE
C2CF C6    02     ADGK  LDAB   #2
C2D1 BD    C9BE         JSR    PSPACE      2 SPACES
C2D4 86    23           LDAA   #$23        "#"
C2D6 BD    CD18         JSR    PUTCHR
C2D9 BE    C84F         LDX    #FCB+15     VOL#
C2DC 5F                 CLRB               NO SPACES
C2DD BD    CD39         JSR    OUTDEC
C2E0 BD    CD24         JSR    PCRLF
C2E3 7D    C10C         TST    EXPFLG      EXP MODE?
C2E6 27    06           BEQ    ADGK1       SHORT MODE
C2E8 BE    C61E         LDX    #STG2       NAME, TYPE, ECT
C2EB BD    CD1E         JSR    PMSG

C2EE BD    CD24   ADGK1 JSR    PCRLF
C2F1 BE    C855         LDX    #FCB+21     SECTORS LEFT
C2F4 A6    84           LDAA   0,X         MSB
C2F6 B7    C105         STAA   SIZE        SAVE FOR LATER
C2F9 A6    01           LDAA   1,X         LSB
C2FB B7    C106         STAA   SIZE+1
C2FE BE    C840         LDX    #FCB        FCB
C301 86    06           LDAA   #6          OPEN DIRECTORY
C303 A7    84           STAA   0,X         SAVE FOR READ OPEN
C305 BD    0406         JSR    FMS         CALL FMS - DO OPEN
```

```
C308 27    03           BEQ    CAT4        OK
C30A 7E    C53C         JMP    CAT9        ERROR
C30D BE    C840   CAT4  LDX    #FCB        RESET
C310 86    07           LDAA   #7          GET INFO REC
C312 A7    84           STAA   0,X         GET REC
C314 BD    0406         JSR    FMS         CALL FMS - GET REC
C317 27    03           BEQ    CAT4A       OK
C319 7E    C533         JMP    CAT6        ERRORS
C31C BE    C840   CAT4A LDX    #FCB        START OF INFO
C31F A6    04           LDAA   4,X         FETCH 1ST CHR
C321 81    FF           CMPA   #$FF        DELETED?
C323 26    09           BNE    CAT4C       NOT DELETED!
C325 7D    C113         TST    DELFG       LIST DELETED FILE?
C328 27    E3           BEQ    CAT4        NO, SKIP IT
C32A 86    3F           LDAA   #'?         FILL IN DEL CHR
C32C A7    04           STAA   4,X         SAVE FOR LATER
C32E A6    0F     CAT4C LDAA   15,X        ATTRIBUTES
C3 0 84    10           ANDA   #$10        CAT PROTECTED?
C332 27    05           BEQ    NCPRT       NOT CAT PROTECTED
C334 7D    C114         TST    CATFG       CAT PROT OPTION?
C337 27    04           BEQ    CAT4        NO, SKIP IT
C339 A6    04     NCPRT LDAA   4,X         REFETCH
C33B 81    00           CMPA   #$00        LAST ENTRY?
C33D 26    33           BNE    CAT4B       WAIT
C33F BD    CD24         JSR    PCRLF
C342 BE    C65C         LDX    #STG5       SECTORS LEFT
C345 BD    CD1E         JSR    PMSG
C348 BE    C105         LDX    #SIZE       SECTORS LEFT
C34B 5F                 CLRB               NO SPACES
C34C BD    CD39         JSR    OUTDEC
C34F BD    CD24         JSR    PCRLF
C352 7D    C10F         TST    DEFFG       DEFAULT MODE?
C355 26    13           BNE    NCPRT1      NO DEFAULT DRV
C357 7D    C10E         TST    AUTFG       ALREADY DONE?
C35A 27    0E           BEQ    NCPRT1      DONE
C35C 7A    C10E         DEC    AUTFG       NEXT DRV (AUTO MODE)
C35  7D    C10E         TST    AUTFG       DRV  DONE?
C362 26    03           BNE    CAT0D       NOT YET
C364 7E    C539         JMP    CAT6A       FLEX
C367 7E    C22D   CAT0D JMP    CAT2        AGAIN
C36A 7D    C112  NCPRT1 TST    DRVFG       DONE?
C36D 26    F8           BNE    CAT0D       NO YET!
C36F 7E    C539         JMP    CAT6A       DONE
C372 7D    C100   CAT4B TST    MATFLG      MATCH STRING?
C375 27    67           BEQ    MAT3        NO MATCH STRING REQUIRED
C377 BE    C119         LDX    #STGPTR     1ST PNTR
C37A BF    C117         STX    IDXTMP      RESET PNTR

                  *AT THIS POINT WE MUST CHECK THE FILE NAME
                  *AGAINST THE STRING IN THE INPUT BUFFER STARTING
                  *AT THE ADDRESS CONTAINED IN LOCATION 'IDXTMP'....

C37D BE    C844   MAT4A LDX    #FCB+4      SOURCE COMPARE (NAME)
C380 BF    C107         STX    PTRS        SAVE
C383 BE    C117         LDX    IDXTMP      ADDR OF PNTR
C386 A6    84           LDAA   0,X         MSB
C388 B7    C109         STAA   PTRD
C38B A6    01           LDAA   1,X         LSB
C38D B7    C10A         STAA   PTRD+1
C390 BE    C109   MAT4  LDX    PTRD        DEST
C393 A6    84           LDAA   0,X         FETCH CHR
C395 81    2E           CMPA   #'.         EXT?
C397 27    35           BEQ    EXT1        EXT TIME!
C399 81    30           CMPA   #'0         ZERO?
C39B 25    41           BCS    MAT3        MATCH OK
C39D 1F    8940         TAB                SAVE

C3A0 30    01           INX                NEXT CHR
C3A2 BF    C109         STX    PTRD        UPDATE
C3A5 BE    C107         LDX    PTRS        SOURCE
C3A8 A6    84           LDAA   0.X         FETCH COMP CHAR
C3AA 30    01           INX                NEXT CHR
C3AC BF    C107         STX    PTRS        UPDATE
C3AF 34    04 A1ED      CBA                SAME?
C3B3 27    06           BEQ    MAT4        MATCH-CONTINUE
C3B5 BE    C117         LDX    IDXTMP      SET PNTR
C3B8 30    01           INX                NEXT ONE
C3BA 30    01           INX
C3BC BF    C117         STX    IDXTMP      UPDATE
C3BF A6    84           LDAA   0,X         FETCH
C3C1 81    FF           CMPA   #$FF        EOT?
C3C3 26    B8           BNE    MAT4A       NO
C3C5 BE    C119         LDX    #STGPTR     1ST PNTR
C3C8 BF    C117         STX    IDXTMP      RESET
C3CB 7E    C30D         JMP    CAT4        NEXT FILE
C3CE BE    C84C   EXT1  LDX    #FCB+12     EXTENSION AREA
C3D1 BF    C107         STX    PTRS        SAVE
C3D4 BE    C109         LDX    PTRD        STRING PNTR
C3D7 30    01           INX                SKIP OVER '.'
C3D9 BF    C109         STX    PTRD        SAVE
C3DC 20    B2           BRA    MAT4        GO MATCH EXT!
C3DE 7D    C10C   MAT3  TST    EXPFLG      EXP MODE?
C3E1 27    2E           BEQ    SHT1        SHORT MODE
C3E3 7C    C111         INC    FILCNT+1    FILE CNTR
C3E6 B6    C111         LDAA   FILCNT+1    FETCH CNTR
C3E9 81    DA           CMPA   #$0A        >9?
C3EB 24    09           BCC    NNSP        NO SPACE
C3ED 34    02           PSHA
C3EF 86    20           LDAA   #$20
C3F1 BD    CD18         JSR    PUTCHR
C3F4 35    02           PULA
C3F6 BE    C11D   NNSP  LDX    #FILCNT     POINT TO FILE #
C3F9 5F                 CLRB               SUPPRESS SPACES
C3FA BD    CD39         JSR    OUTDEC      PRINT FILE #
C3FD C6    02           LDAB   #2          2 SPACES
C3FF BD    C5BE         JSR    PSPACE
C402 BE    C843         LDX    #FCB+3      DRV #
```

```
C405 A6   84              LDAA   0,X        FETCH DRIVE
C407 8B   30              ADDA   #$30       ASCII OFFSET
C409 BD   CD18            JSR    PUTCHR     PRINT DRV#
C40C 86   2E              LDAA   #'.        PERIOD
C40E BD   CD18            JSR    PUTCHR
C411 8E   C844   SHT1     LDX    #FCB+4     FILE NAME
C414 C6   08              LDAB   #8         8 CHRS
C416 A6   84     LP1      LDAA   0,X        FETCH CHR
C418 81   00              CMPA   #$00       NOP?
C41A 26   02              BNE    LP1A       SKIP
C41C 86   20              LDAA   #$20       MAKE IT A SPACE
C41E BD   CD18   LP1A     JSR    PUTCHR     PRIN  NAME CHR
C421 30   01              INX               NEXT CHR
C423 5A                   DECB
C424 26   F0              BNE    LP1        NEXT CHR
C426 86   2E              LDAA   #'.
C428 BD   CD18            JSR    PUTCHR
C42B C6   03              LDAB   #3         3 CHRS
C42D A6   84     LP       LDAA   0,X        FETCH EXT CHR
C42F BD   CD18            JSR    PUTCHR
C432 30   01              INX
C434 5A                   DECB
C435 26   F6              BNE    LP         NEXT CHR
C437 7D   C10C            TST    EXPFLG     EXP MODE?
C43A 26   21              BNE    LP3        EXP MODE!
C43C 7D   C116            TST    OLDFG      OLD CAT FORMAT?
C43F 26   1C              BNE    LP3        YES!
C441 B6   C108   LP2A     LDAA   CLMCNT     WHICH CLM?
C444 4C                   INCA              OFFSET
C445 B1   C115            CMPA   CLMLIM     TIME FOR CR/LF?
C448 26   08              BNE    LP4        NO, BUT RELOOP
C44A BD   CD24            JSR    PCRLF
C44D 7F   C108            CLR    CLMCNT     RESET
C450 20   08              BRA    LP4A       SKIP
C452 7C   C108   LP4      INC    CLMCNT     CLM+1
C455 C6   03              LDAB   #3         3 SPACES
C457 BD   C5BE            JSR    PSPACE
C45A 7E   C300   LP4A     JMP    CAT4       NEXT FILE NAME
C45D 8E   C855   LP3      LDX    #FCB+21    FILE SIZE
C460 C6   FF              LDAB   #$FF       LEADING SPACES
C462 BD   CD39            JSR    OUTDEC     SECTOR CNT (DEC)
C465 C6   02              LDAB   #2
C467 BD   C5BE            JSR    PSPACE     2 SPACES
C46A 7D   C10C            TST    EXPFLG     EXP MODE?
C46D 27   5C              BEQ    MTH1       NO, OLD SHORT FORM
C46F 8E   C85A            LDX    #FCB+26    DAT
C472 7F   C103            CLR    STORE      CLR STORAGE
C475 A6   84              LDAA   0,X        FETCH DAY
C477 B7   C104            STAA   STORE+1    SAVE IT
C47A 81   0A              CMPA   #$0A       >9?
C47C 24   09              BCC    NSP        NO SPACE
C47E 34   02              PSHA
C480 86   20              LDAA   #$20
C482 BD   CD18            JSR    PUTCHR
C485 35   02              PULA
C487 8E   C103   NSP      LDX    #STORE     POINT TO DAY
C48A 5F                   CLRB              SUPPRESS SPACES
C48B BD   CD39            JSR    OUTDEC     PRINT DAY
C48E 86   2D              LDAA   #'-
C490 BD   CD18            JSR    PUTCHR
C493 8E   C859            LDX    #FCB+25    MONTH
C496 A6   84              LDAA   0,X        FETCH MONTH
C498 8E   C5D5            LDX    #TABLE     MONTH TABLE
C49B 4A          NXTMNT   DECA              DEC MONTH
C49C 27   0A              BEQ    FOUND      GOT THE MONTH
C49E 30   01     STEP     INX
C4A0 6D   84              TST    0,X        END OF ENTRY?
C4A2 26   FA              BNE    STEP       NEXT STEP
C4A4 30   01              INX               START OF NEXT ENTRY
C4A6 20   F3              BRA    NXTMNT     NEXT MONTH
C4A8 A6   84     FOUND    LDAA   0,X        FETCH CHR
C4AA 27   08              BEQ    MTH        NOP IF '00'
C4AC BD   CD18            JSR    PUTCHR     PT IT
C4AF 30   01              INX               NEXT CHR
>C4B1 7E  C4A8            JMP    FOUND
C4B4 86   2D     MTH      LDAA   #'-
C4B6 BD   CD18            JSR    PUTCHR
C4B9 8E   C85B            LDX    #FCB+27    YEAR
C4BC A6   84              LDAA   0,X        FETCH YEAR
C4BE 7F   C103            CLR    STORE
C4C1 B7   C104            STAA   STORE+1    SAVE IT
C4C4 5F                   CLRB              SUPPRESS ZEROS
C4C5 8E   C103            LDX    #STORE     POINT TO YEAR
C4C8 BD   CD39            JSR    OUTDEC
C4CB 8E   C84F   MTH1     LDX    #FCB+15
C4CE C6   03              LDAB   #3
C4D0 BD   C5BE            JSR    PSPACE     3 SPACES
C4D3 A6   84              LDAA   0,X        FETCH ATTRIBUTES
C4D5 84   80              ANDA   #$80       'WRITE' PROT?
C4D7 27   04              BEQ    PRT1A      NO
C4D9 86   57              LDAA   #'W        WRI E PROT
C4DB 20   02              BRA    PRT1       SKIP
C4DD 86   2A     PRT1A    LDAA   #'*
C4DF BD   CD18   PRT1     JSR    PUTCHR     PT WRITE PROT
C4E2 A6   84              LDAA   0,X        ATTRIB
C4E4 84   40              ANDA   #$40       'DELETE' PROT?
C4E6 27   04              BEQ    PRT A      NO
C4E8 86   44              LDAA   #'D        DELETE PROT
C4EA 20   02              BRA    PRT2       SKIP
C4EC 86   2A     PRT2A    LDAA   #'*
C4EE BD   CD18   PRT2     JSR    PUTCHR     PT DELETE PROT
C4F1 A6   84              LDAA   0,X        ATTRIB
C4F3 84   20              ANDA   #$20       'READ' PROT?
C4F5 27   04              BEQ    PRT3A      NO
C4F7 86   52              LDAA   #'R        READ PROT
C4F9 20   02              BRA    PRT3       SKIP
C4FB 86   2A     PRT3A    LDAA   #'*
```

```
C4FD BD   CD18   PRT3     JSR    PUTCHR
C500 A6   84              LDAA   0,X        ATTRIB
C502 84   10              ANDA   #$10       'CATALOG' PROT?
C504 27   04              BEQ    PRT4A      NO
C506 86   43              LDAA   #'C        CAT PROT
C508 20   02              BRA    PRT4       SKIP
C50A 86   2A     PRT4A    LDAA   #'*

C50C BD   CD18   PRT4     JSR    PUTCHR
C50F 7D   C10C            TST    EXPFLG     EXP MODE?
C512 26   03              BNE    PRT4D      YES!
C514 7E   C441            JMP    LP2A       NO, OLD SHORT FORM
C517 30   01     PRT4D    INX
C519 30   01              INX               START ADDR OF FILE
C51B C6   02              LDAB   #2
C51D BD   C5BE            JSR    PSPACE     2 SPACES
C520 BD   CD45            JSR    OUTADR     TRACK-SECTOR
C523 30   01              INX               END ADDR OF FILE
C525 C6   02              LDAB   #2
C527 BD   C5BE            JSR    PSPACE     2 SPACES
C52A BD   CD45            JSR    OUTADR     TRACK-SECTOR
C52D BD   CD24   PRT4B    JSR    PCRLF      CRLF
C530 7E   C300            JMP    CAT4       REPEAT
C533 A6   01     CAT6     LDAA   1,X        GET ERROR STATUS
C535 81   08              CMPA   #8         EOF ERROR?
C537 26   03              BNE    CAT9
C539 7E   CD03   CAT6A    JMP    WARMS
C53C BD   CD3F   CAT      JSR    RPTERR     REPORT ERR
C53F 7E   CD03            JMP    WARMS
C542 8E   C66C   CAT6B    LDX    #STG6      SYNTAX ERR
C545 BD   CD1E            JSR    PMSG
C548 20   EF              BRA    CAT6A      ABORT


                 *<<<<<<<<<<<<<<<<<< OPTION >>>>>>>>>>>>>>>>>>>>>
                 *TEST FOR OPTION CODES AND SET FLAGS ACCORDINGLY

C54A 81   2B     OPTION   CMPA   #OPTCHR    OPTION LEAD IN CHR?
C54C 27   03              BEQ    OPT2       YES
C54E 1C   FE     OPT1     CLC               NO SYNTAX ERROR
C550 39                   RTS
C551 BD   CD27   OPT2     JSR    NXTCH      GET OPTION
C554 81   2C              CMPA   #',        COMMA?
C556 27   F6              BEQ    OPT1       DONE
C558 81   20              CMPA   #$20       SPACE?
C55A 27   F2              BEQ    OPT1       DONE
C55C 81   0D              CMPA   #$D        CR?
C55E 27   EE              BEQ    OPT1       DONE
C560 81   45              CMPA   #EXPCHR    EXP MODE?
C562 26   05              BNE    OPT3       NO
C564 7C   C10C            INC    EXPFLG     EXP MODE
C567 20   E8              BRA    OPT2       MORE?
C569 81   44     OPT3     CMPA   #DELCHR    CAT DEL FILES?
C56B 26   05              BNE    OPT4       NO
C56D 7C   C113            INC    DELFG      CAT DEL FILES
C570 20   DF              BRA    OPT2       MORE?
C572 81   50     OPT4     CMPA   #PROCHR    CAT PROT FILES?
C574 26   20              BNE    OPT5       MORE?
C576 86   59              LDAA   #PASS      PASSWORD MECHANISM ON?
C578 81   59              CMPA   #'Y        YES?
C57A 26   15              BNE    OPT4A      IGNORE PASSWORD
C57C BD   CD27            JSR    NXTCH
C57F 81   44              CMPA   #KEY       MATCH? (1ST CHR)
C581 26   38              BNE    OPT6       NO MATCH
C583 BD   CD27            JSR    NXTCH
C586 81   4F              CMPA   #KEY1      MATCH? (2ND CHR)
C588 26   31              BNE    OPT6       NO MA CH
C58A BD   CD27            JSR    NXTCH
C58D 81   47              CMPA   #KEY2      MATCH? (3RD CHR)
C58F 26    A              BNE    OPT6       NO MATCH
C591 7C   C114   OPT4A    INC    CATFG      PT CAT PROT FILES
C594 20   BB              BRA    OPT2       MORE
C596 81   43     OPT5     CMPA   #CLMCHR    CLM CNT CHANGE?
C598 26   21              BNE    OPT6       SYNTAX ERROR
C59A BD   CD27            JSR    NXTCH      GET CLM CNT
C59D 25   1C              BCS    OPT6       NOT ALPHA-NUM, ERROR
C59F 81   3A              CMPA   #$3A       TOO LARGE?
C5A1 24   18              BCC    OPT6       ERROR
C5A3 81   30              CMPA   #$30       ZERO?
C5A5 27   0A              BEQ    OPT5A      OLD FORM
C5A7 84   0F              ANDA   #$0F       STRIP ASCII
C5A9 B7   C115            STAA   CLMLIM     SAVE FOR LATER
C5AC 7F   C116            CLR    OLDFG      OLD FORM OFF
C5AF 20   A0              BRA    OPT2       MORE?
C5B1 86   01     OPT5A    LDAA   #1         1 CLM
C5B3 B7   C115            STAA   CLMLIM
C5B6 7C   C116            INC    OLDFG      OLD CAT FORMAT
C5B9 20   96              BRA    OPT2       MORE?
C5BB 1A   01     OPT6     SEC               ERROR FLAG
C5BD 39                   RTS

                 *<<<<<<<<<<<<<<< PSPACE >>>>>>>>>>>>>>>>>>
                 *PRINT # OF SPACE SET IN 'B' ACC
C5BE 86   20     PSPACE   LDAA   #$20       SPACE
C5C0 BD   CD18            JSR    PUTCHR     PRINT SPACE
C5C3 5A                   DECB
C5C4 26   F8              BNE    PSPACE     MORE
C5C6 39                   RTS


                 *<<<<<<<<<<<<<<<<<<< PMSG1 >>>>>>>>>>>>>>>>>
                 *PRINT STRING W/O A CR-LF
C5C7 A6   84     PMSG1    LDAA   0,X        FETCH CHR
C5C9 81   04              CMPA   #4         EOT?
C5CB 27   07              BEQ    PMSG2      DONE
C5CD BD   CD18            JSR    PUTCHR     PT IT
C5D0 30   01              INX               NEXT CHR
C5D2 20   F3              BRA    PMSG1
C5D4 39          PMSG2    RTS
```

FREE DISKETTE WITH EVERY $50 PURCHASE

☎ TOLL FREE
1-800-338-6800
For Ordering

TELEX 558 414 PVT BTH

SOUTH EAST MEDIA

5900 Cassandra Smith Rd.
Hixson, TN 37343
for information
call (615) 842-4601

CoCo OS-9™ FLEX™
SOFTWARE

!!! SPECIALS !!!

"Tear End CLEARANCE" --- While they Last

FLEX Software ---------

TSC "FLEX Utilities"	was $75.00,	NOW only $60.00
TSC "Sort Merge"	was $75.00,	NOW only $60.00
TSC "6809 BASIC"	was $75.00,	NOW only $60.00
TSC "Extended BASIC"	was $100.00,	NOW only $85.00
TSC "DeBug"	was $75.00,	NOW only $60.00
TSC "FLEX Diagnostics"	was $75.00,	NOW only $60.00
TSC "Text Processing System"	was $75.00,	NOW only $60.00
TSC "68000 Cross Assembler"	was $250.00,	NOW only $220.00

LUCIDATA "TEKPAK" (FLEX9, 5 1/4") -- A Pascal implementation of the Tektronix 40xx Terminal Control System, **with Pascal SOURCE**. The Manual includes a discussion of how to utilize this package in the graphical library in implementing Vector Drawing, Point Plotting, etc., up through Windowing and Clipping concepts.
Normal Price, $100.00; **NOW** only **$85.00**

OS-9 Software ------

MICROWARE "OS-9 File Handler Toolbox";	NOW only $70.00
MICROWARE "Relocating Macro Assembler";	NOW only $170.00

!!! NEW NEW NEW !!!

Computer Systems Consultants, Inc.

CMODEM TELECOMMUNICATIONS PROGRAM
Menu-Driven; supports Dumb-Terminal Mode, Upload and Download in non-protocol mode, and the CP/M "MODEM7" Christensen protocol mode to enable communication capabilities for almost any requirement. Written in "C".

FLEX, CCF, OS-9, UniFLEX; with complete Source - **$100.00**
without Source - **$50.00**

M-E-M -- TRUE (not Macros) CROSS ASSEMBLERS
Use your 6809-Based Computer System for developing Software for 1802/5, 6800/01/03/11, 6804, 6805, 6809, 6502/3, 8080/5, 8048, 8051, Z-80, and 68000 Systems. Provides the Assembler Language and Listings normally used on the target Systems. Written in "C"; produces Motorola S-Text for machine independence.

FLEX, CCF, OS-9, UniFLEX
each - **$50.00**
any 3 - **$100.00**
the complete set (including the C Source) - **$200.00**

LLOYD I/O

K-BASIC -- A "Native Code" **BASIC Compiler**
Level 1 K-BASIC supports sequential files, floating point, 3 sizes of integers, string variables, and arrays. The single-pass compiler compiles to Assembly Language Source Code (which may be assembled by the included OSM Assembler, or by the CRASMB Cross Assemblers). Conditional assembly is used to reduce the size of the run time package. (See **Review in Oct. '84 Issue of '68' Micro Journal.**)

FLEX, CCF, OS-9 Compiler **with** OSM Assembler - **$199.00**

OSM -- Extended 6809 Macro Assembler
Provides local labels, Motorola S-records, and Intel Hex records. Also generates OS-9 Memory modules under FLEX, allowing the maintenance of source code programs for both DOS's on one System.

FLEX, CCF, OS-9 **$99.00**

CRASMB -- 8-Bit Macro Cross Assembler
Same features as OSM, cross-assembles to 6800/2/8, 6801/3, 6804, 6805, 6809, 6811, 6502, 1802, 8048, 8080/5, Z-8, Z-80. Fully supports the target chip's standard mnemonics and addressing modes.

FLEX, CCF, OS-9 full package -- **$399.00**

CRASMB 16.32 -- Cross Assembler for the **68000**
Same features as 8-Bit Cross Assemblers above
$249.00

Compusense Ltd.

CRUNCH COBOL -- COBOL Compiler
This COBOL Compiler supports a large subset of ANSII Level 1 COBOL with many of the useful Level 2 features. Full support of the FLEX File Structures is implemented including Random Files and the ability to process Keyed Files. Large programs can be segmented and linked at runtime, or implemented as a set of overlays. The System requires 56K and CAN be run with a single Disk System.

FLEX, CCF Normally $199.00
Special Introductory Price (while they last) -- $99.95

ASSEMBLERS

Southeast Media

ASTRUK09
A "Structured Assembler for the 6809" which requires the **TSC Macro Assembler**. Allows direct use of structured statements such as IF, ELSE, DO, REPEAT, etc., and provides indented level formatting of the listing so that the structure is apparent. **Re. '68' Micro Journal, Sept. '83** (program was called "STASM09"; has been renamed due to conflicts).

A User reports
"... I'm very pleased and am now writing almost exclusively in (ASTRUK09). I've selected it over --- for all future systems development... As (one) of my early evaluations, I rewrote a rather elaborate routine originally done in assembly. Out of the 1000 bytes of code generated, the (ASTRUK09) version used only 20 more bytes than the original. --- could not handle this program since it uses triple-precision fixed point arithmetic... I have a large body of code already written that is incompatible with --- constructs. No problem with (ASTRUK09) and the structure sure helps in understanding the logic!"

F, CCF - **$99.95**

TSC

Macro Assembler
The FLEX **STANDARD** Assembler. **Special** -- F,CCF **$35.00**

Relocating Assembler w/Linking Loader
Use with many of the C and Pascal Compilers. F,CCF **$150.00**

Great Plains Comp. Co.

RMAC
Relocating, Recursive-Macro Assembler and **Linking Loader.**
F,CCF **$120.00**; w/Source **$240.00**

OmegaSoft

CRALLI
Relocating Assembler and **Linking Loader**
F,CCF **$125.00**; for One Year Maint., add **$50.00**

Windrush Micro Systems
MACE, by Graham Trott. F,CCF - **$98.00**

VISA MasterCard

** SHIPPING **
Add 2% U.S.A.
(min. $2.50)
Add 5% Surface Foreign
10% Air Foreign

☎ TOLL FREE
1-800-338-6800
SOUTH EAST MEDIA
5900 Cassandra Smith Rd
Hixson, TN 37343
info (615) 842-4601
CoCo OS-9™ FLEX™
SOFTWARE

Availability Legends ---
F = FLEX, CCF = Color Computer FLEX
O = OS-9, CCO = Color Computer OS-9
U = UniFLEX
CCD = Color Computer Disk
CCT = Color Computer Tape

*FLEX is a trademark of Technical Systems Consultants
**OS9 is a trademark of Microware

!!! Please Specify Your Operating System & Disk Size !!!

DISASSEMBLERS

Computer Systems Consultants

SUPER SLEUTH

Computer Systems Consultants Super Sleuth is a "Time Tested", reliable, PROVEN Disassembler that has gained acceptance through out the SS-50 Bus Community as an extremely POWERFUL, INTERACTIVE, Software Tool. The Super Sleuth Software Package consists of 3 Programs; SLEUTH (the Disassembler), CHGNAM (used to globally Change Labels to a meaningful Name), and XREF (a Cross Reference Generator for Source Code Files). SLEUTH will Disassemble Memory Resident 6809 Code and 6800, 6801, 6802, 6803 (the "Baby CoCo"), 6805, 6808, 6809, and 6502 (Apple, Atari, Commodore, etc.) Binary Disk Files. (See Aug. '83 '68' Micro Journal "Color Users Notes" Column for a full Review.)

Color Computer	SS-50 Bus (all w/ Source)
CCO (32K Req'd)	
Obj. Only $49.00	F, $99.00
CCF, Obj. Only $50.00	U, $100.00
CCF, w/Source $99.00	O, $101.00
CCO, Obj. Only $50.00	

ALL Computer Systems Consultants Software
runs on the Color FLEX Systems
ALL in stock
call 800-338-6800
for IMMEDIATE DELIVERY

Computer Systems Center

DYNAMITE +

An "easy to use", powerful Disassembler for Disk Resident 6809 and 6800 Binary Files. Allows the development of a "Control File" of various Program "Boundaries" during successive disassemblies; can use a Label File which automatically replaces a Hex Location with a Label Name; includes an XREF Utility; etc. Label Files provided for Mini-FLEX, FLEX2, FLEX9, Color Computer (for use with Color FLEX Systems), etc. OS-9 Version includes special OS-9 options.

CCF, Obj. Only	$100.00	CCO, " "	$150.00
F, " "	$100.00	O, " "	$150.00
U, " "	$300.00		

COMPILERS and DECOMPILERS

6809 "Structured" Assembly Lang. Compilers

Windrush Micro Systems

PL/9

By Graham Trott. A combination Editor/Compiler/Debugger, all in ONE PACKAGE; provides a totally INTERACTIVE Program Development Cycle. The Single-Pass Compiler supports large Symbol Names; Variable Types; Pointers; Control Structures (similar to 'C' or 'Pascal'); Stack, A- ,B-, and D-Register manipulation; etc. The Source-Oriented Trace/Debugger provides Single Stepping, Breakpointing, etc. An excellent Software Development Tool which provides for the maximum utilization of the power of the 6809.

F, CCF - $198.00

Whimsical Developments

WHIMSICAL

Need the Ease of Design and Maintainability of "Structured Programming" **AND** the Speed and Control of Assembly Language? Then WHIMSICAL was designed for you! This Single Pass, Recursive Descent Compiler provides the tool for developing simple Utilities to MAJOR Systems in Assembly Language. Supports 3 "Lex" Levels which allow one level of Procedure nesting, or more within "Modules". It is easy to develop programs written for other machines since you are working at the Assembly Language level. Features unified, user-defined I/O; produces ROMable, relocatable, recursive, re-entrant Code; Structured style and statements with Procedures and Modules; supports Byte and Double-Byte primitives with 3 types of Integers (up to 32 bit), Char and Boolean, and unlimited sized Arrays (vectors only); Interrupt handling; unlimited length Variable Names; Variable Initialization (defaults to $00); Include "Source File" directive; Conditional compiling; direct Code insertion; control of the Stack Pointer; etc. To quote Ron Anderson in his comments about WHIMSICAL in the Sept. '83 Issue of **'68' Micro Journal** that, except for the lack of floats, "...., **I have to give this one VERY high rating,** ...". It is a FAST Compiler which produces FAST Code (his "Primes" Benchmark ran at 9 secs. on a 2 MHz System).

F and CCF - $195.00

FREE DISKETTE WITH EVERY $50 PURCHASE

☎ TOLL FREE
1-800-338-6800
For Ordering

TELEX 558 414 PVT BTH

SOUTH EAST MEDIA

5900 Cassandra Smith Rd.
Hixson, TN 37343

for information
call (615) 842-4601

CoCo OS-9™ FLEX™
SOFTWARE

'C' COMPILERS

Windrush Micro Systems

C Compiler

By James McCosh. Full featured C Compiler for the FLEX Operating System (lacking ONLY "bit-fields"), including an Assembler. Requires the TSC Relocating Assembler IF the user wishes to implement his own Libraries.

F and CCF - $295.00

Introl

C Compiler

A full-featured C, streamlined for the 6809. Generates very efficient object code. Output "benchmarks" close to 10MHz 68000 in 8 Bit Operations; 1.5 times faster than a 4 MHz Z80 when using a 2MHz 6809 System (Re. p 43, "68" Micro Journal, May '83). Floats, etc.

F, CCF, and O	$375.00
U	$425.00
One Year Maint.	$100.00

PASCAL COMPILERS

TSC

PASCAL Compiler

Native Code Compiler (UCSD Oriented).

F and CCF - $200.00

Lucidata

PASCAL Compiler

P-Code Compiler (ISO Standard). Designed especially for Microcomputer Systems; Run-time System checks available resources for each task, allowing operation on even minimal computer systems. Allows linkage to Assembler Code for maximum flexibility.

F and CCF 5"	$190.00
F 8"	$205.00

OmegaSoft

PASCAL Compiler

For the PROFESSIONAL; ISO Based, Native Code Compiler. Primarily for Real-Time and Process Control applications. Use custom I/O devices in place of the Pascal INPUT and OUTPUT; Long Int. (32 Bit); Dynamic length strings; Interrupt processing, ROM-able, PIC, Re-Entrant Code, etc. POWERFUL! Includes Source for the Symbolic Debugger, Runtime, and several Utilities. Requires a "Motorola Compatible" Relocating Assembler and Linking Loader.

F and CCF	$425.00
One Year Maint.	$100.00

DECOMPILERS

Southeast Media

DUB (A UniFLEX "basic" De-Compiler)

Re-Create a Source Listing from UniFLEX Compiled basic Programs. Easy to Use; works w/ ALL Versions of UniFLEX basic; Output to Disk or Terminal. Time TESTED and PROVEN; SOLID!

U - $219.95

VISA MasterCard

** SHIPPING **
Add 2% U.S.A.
(min. $2.50)
Add 5% Surface Foreign
10% Air Foreign

☎ TOLL FREE
1-800-338-6800
For Ordering

SOUTH EAST MEDIA
5900 Cassandra Smith Rd.
Hixson, TN 37343
info (615) 842-4601

CoCo OS-9™ FLEX™
SOFTWARE

Availability Legends —

F = FLEX,	CCF = Color Computer FLEX
O = OS-9,	CCO = Color Computer OS-9
U = UniFLEX	
CCD = Color Computer Disk	
CCT = Color Computer Tape	

™FLEX is a trademark of Technical Systems Consultants
™OS9 is a trademark of Microware

!!! Please Specify Your Operating System & Disk Size !!!

FREE DISKETTE WITH EVERY $50 PURCHASE

☎ TOLL FREE
1-800-338-6800
For Ordering

TELEX 558 414 PVT BTH

SOUTH EAST MEDIA

5900 Cassandra Smith Rd.
Hixson, TN 37343
for information
call (615) 842-4601

CoCo OS-9™ FLEX™
SOFTWARE

UTILITIES

Southeast Media

BasicO9 XRef

This **BasicO9 Cross Reference Utility** is a BasicO9 Program which will produce a "pretty printed" listing with each line numbered, followed by a complete cross referenced listing of all variables, external procedures, and line numbers called. Also included is a **Program List Utility** which outputs the listing without the overhead of building the cross reference table, which allows it to run considerably faster when only a "pretty printed" listing with line numbers is desired. Requires BasicO9 or RunB for operation.

```
72    [illegible]
73       GET #InPath,[illegible]
74       GET #InPath,[illegible] \ GET #InPath,[illegible]
75       SEEK #InPath,[illegible]
76       [illegible]
77       REPEAT
78          GET #InPath,char
79          [illegible]
80       UNTIL char=127
81    UNTIL [illegible]
82    RETURN

[illegible]    3   26   60   70   76
Name           3   19   30   81
OutPath        4   51   54   56
char           4   28   29   38   33   40   41   42   44   45   46
              70   79   80
found          4   72   60   33
   10          9   11
   20         11   13
```

O and CCO - Obj. Only -- $39.95
O and CCO - w/ Source - $79.95

Southeast Media

OS-9 VDisk

Give your OS-9 Level I System the speed of memory access that can be several orders of magnitude over your present floppy disk drive. Use that Extended Memory capability of your SWTPC or Gimix CPU card (or any other that has the same format DAT). The size of the Virtual Disk is completely variable in whole increments of 4K up to 960K, which is all that these systems can address beyond the base page that OS-9 Level I uses. By putting all of your CMDS Directory on your Virtual Disk, you can have the fastest execution speed possible (next to eating up System Memory with all of them). You can also set up high speed inter-process communications via random virtual disk files and not eat up valuable system memory with pipe buffers. Some Assembly Required - Level I ONLY.

O, obj. only - $79.95
w/ Source - $149.95

Southeast Media

O-F

---- OS/9 to FLEX - FLEX to OS/9 ----

Finally; the barrier has been removed between OS/9 and FLEX formatted disks! Now you can READ **from**, and WRITE **to**, a Single Sided 5" or 8" FLEX diskette from OS-9 with O-F. O-F is a new and unique program, written in BASIC09 (with Source), that performs the following functions;

REFORMAT: A BASIC09 Program that reformats a chosen amount of an OS-9 disk to FLEX Format so it can be used normally by FLEX.

FLEX: A BASIC09 Program that does the actual **read** or **write** function to the special O-F Transfer Disk, all selected from a user-friendly menu. Functions provided include reading the FLEX Directory, Deleting FLEX Files, Copying both directions, etc. All selections are interactive and complete, including all necessary prompts to the operator.

FLEX users can read, write and use the special disk as any other FLEX disk, provided the FLEX directory is not allowed to continue beyond track zero (too many files).

O - $79.95

Southeast Media

COPYMULT

--- Copy LARGE Disks to several smaller disks ---

The following FLEX utilities allow the backup of ANY size disk to any SMALLER size diskettes (Winchester to 8's or 5's, 8" to 5's, etc.). By simply inserting diskettes as requested by COPYMULT, a large disk system may be downloaded to your present floppy disk system, any size. No need to fiddle with directory deletions or any of the other tedious operations that must be done using the normal copy routines.

COPYMULT.CMD understands normal "copy" syntax and always keeps up with files already copied by maintaining directories for both host and receiving disk system, eliminating hours of tedious keyboard entries and other time consuming cleanup chores.

BACKUP.CMD is a special program that downloads "random" type files, any size.

RESTORE.CMD a special program to restructure copied "random" files for copying, or recopying back to the host system.

FREELINK.CMD a "bonus" utility that "relinks" the free chain of floppy or hard disk thereby eliminating fragmentation.

Completely documented source files included.

ALL 4 Programs (FLEX, 8" or 5") $99.50

Southeast Media

XDATA

A COMMUNICATION Package

for the UniFLEX Operating System

Allows UniFLEX Based Systems to Transmit and Receive files to and from other Computer Systems via Modem. Use with CP/M, Main Frames, other UniFLEX Systems, etc.

- -- Verifies Transmission integrity using checksum or CRC
- -- Automatically Re-Transmits bad blocks
- -- Transmits data in 128 byte blocks

U - $299.99

Lucidata

PASCAL UTILITIES

Requires LUCIDATA Pascal ver 3.

XREF -- produce a Cross Reference Listing of any text; oriented to Pascal Source.

F and CCF - $25.00

INCLUDE -- allows the inclusion of other Files in a Source Text; has unlimited nesting capabilities. Also allows Binary File inclusions.

F and CCF - $25.00

PROFILER -- produces an Indented, Numbered, "Structogram" of a Pascal Source Text File. Allows viewing the overall structure of large programs, and provides clues as to the integrity of the program. Supplied as Source Code; requires compilation.

F and CCF - $25.00

VISA

MasterCard

** SHIPPING **
Add 2% U.S.A.
(min. $2.50)
Add 5% Surface Foreign
10% Air Foreign

☎ TOLL FREE
1-800-338-6800
For Ordering
SOUTH EAST MEDIA
5900 Cassandra Smith Rd
Hixson, TN 37343
info (615) 842-4601
CoCo OS-9™ FLEX™
SOFTWARE

*FLEX is a trademark of Technical Systems Consultants
**OS9 is a trademark of Microware

Availability Legends --

F = FLEX,	CCF = Color Computer FLEX
O = OS-9,	CCO = Color Computer OS-9
U = UniFLEX	
CCD = Color Computer Disk	
CCT = Color Computer Tape	

!!! Please Specify Your Operating System & Disk Size !!!

Lucidata

COPYCAT
Pascal NOT required

Allows reading TSC Mini-FLEX, SSB DOS68, and Digital Research CP/M Disks while operating under FLEX 1.0, FLEX 2.0, or FLEX 9.0 with 6800 or 6809 Systems. COPYCAT will not perform Miracles, but, between the program and the manual, you stand a good chance of accomplishing a transfer. Includes Utilities to List Directories, Copy Files, and convert Text Files when required. Also includes a Utility for investigating Physical Compatibility problems. Programs supplied in **Modular Source Code** (Assembly Language) to make it easier to solve unusual problems.

F and CCF 5" - $50.00
F 8" - $65.00

Computer Systems Consultants

FLEX DISK UTILITIES

Eight (8) different FLEX Utilities that should be a part of every FLEX Users Toolbox; Assembly Language (Source Code):
Copy a File with CRC Errors, so it can possibly be salvaged; **Test Disk** for errors; **Compare two Disks**; a fast **Disk Backup** Program; **Edit Disk Sectors**; **Linearize Free-Chain** on the Disk; print **Disk Identification**; and **Sort and Replace** the Disk Directory (in sorted order).

PLUS

Ten BASIC Programs to:
A **BASIC Resequencer** with EXTRAs over "RENUM"; works with ALL Versions of FLEX BASIC AND the Precompiler, checks for missing label definitions, processes Disk to Disk instead of In Memory.
Compare, Merge, or **Generate Updates** between two BASIC Programs, check **BASIC Sequence Numbers**, compare two unsequenced files, and 5 Programs for establishing a **Master Directory** of several Disks, and sorting, selecting, updating, and printing paginated listings of these files.
A **BASIC Cross-Reference** Program, written in Assembly Language, which provides an X-Ref Listing of the Variables and Reserved Words in **TSC BASIC, XBASIC, and PRECOMPILER BASIC** Programs.
ALL Utilities include **Source** (either BASIC or Source Code). An EXCELLENT Value!

F and CCF - $50.00

BUSINESS
WORD PROCESSING

Windrush Micro Systems

SCREDITOR III

EXTREMELY Powerful Screen-Oriented Editor/Word Processor. Almost 50 different commands; EXCELLENT Documentation (over 300 pages), including a full Tutorial Section to help you learn how to use the system. Features Cursor-based editing, dynamic Screen Formatting (what you see is what you get), Multi-Column display and editing, "decimal align" columns (AND add them up automatically, if wanted), define multiple keystroke macros, even and odd page number headers and footers, imbed printer control codes in text, full justification series of commands, full "help" support, store common command series on disk for future use, etc. Easy "Set-Up" (for example, you just hit the key you want to use for a specific function, such as "cursor up", and the System reads an stores that key - no digging into tech manuals for codes, etc.); use supplied "set-ups", or remap the keyboard to what you are used too. Except for proportional printing, this package will DO IT ALL!

6800 or 6809 FLEX or SSB DOS, OS-9 - $175.00

Southeast Media

SPELLB "Computer Dictionary"
OVER 120,000 words!

No more "Let your fingers do the walking through the Dictionary" while you are entering Text with your favorite Editor or Word Processor. SPELLB is more than just "another Spelling Checker"; it allows you to look up a word from within your Editor or Word Processor so that you **KNOW** it is right WHEN YOU TYPE IT IN with the SPH.CMD Utility (which operates in the FLEX Utility Space). Yes, it ALSO allows you to check and update the Text after you are finished; along with allowing you to ADD WORDS to the Dictionary, "Flag" questionable words in the Text for evaluation later, "View a word in context" before changing or ignoring, etc. SPELLB first checks a "Common Word Dictionary", then the normal Dictionary, then a "Personal Word List", and finally, any "Special Word List" you may have specified. SPELLB also allows the use of **Small Disk Storage** systems.

F and CCF - $129.95

FREE DISKETTE WITH EVERY $50 PURCHASE

☎ TOLL FREE
1-800-338-6800
For Ordering

TELEX 558 414 PVT BTH

SOUTH EAST MEDIA

5900 Cassandra Smith Rd.
Hixson, TN 37343
for information
call (615) 842-4601

CoCo OS-9™ FLEX™
SOFTWARE

Great Plains Computer Co.

STYLOGRAPH

A full-screen oriented WORD PROCESSOR -- (now runs on the **Data-Comp** and FHL Color FLEX Systems; uses the 51 x 24 Display Screens). Full screen display and editing (i.e., what you see is what you get); supports the Daisy Wheel proportional printers.

SPECIAL CCF - $195.00
F and O - $295.00 U - $395.00

SPELL

Fast Computer Dictionary.
F, CCF, OS/9 - $125.00 U - $175.00

MAIL MERGE

Greatly extends the power and flexibility of STYLOGRAPH.
F, CCF, O - $145.00 U - $195.00

Southeast Media

JUST
Text Formatter

JUST, a Text Formatter developed by Ron Anderson, provides numerous features which make it a valuable addition to any FLEX Users Software Library. JUST is designed for formatting Text Output for Dot Matrix Printers and provides many unique features:
- Output the "Formatted" Text to the Display for format analysis and change.
- Output the "Formatted" Text to a Text File for use with the supplied FPRINT.CMD for producing multiple copies of the Text on the Printer INCLUDING IMBEDDED PRINTER COMMANDS (this Utility is very useful at other times also, and worth the price of the program by itself).
- **"User Configurable"** for adapting to other Printers (comes set up for Epson MX-80 with Graftrax); provides for up to ten (10) imbedded "Printer Control Commands", such as Italics on and off, boldface on and off, etc.
- Automatic compensation for a "Double Width" printed line.
- Includes the normal line width, margin, indent, paragraph, space, vertical skip lines, page length, page numbering, centering, fill, justification, etc.
- Use with ANY Editor.
- Supplied with "Structured Source" (Windrush PL/9); easy to see the flow of the program.

F and CCF - $49.95

** SHIPPING **
Add 2% U.S.A.
(min. $2.50)
Add 5% Surface Foreign
10% Air Foreign

*FLEX is a trademark of Technical Systems Consultants
**OS9 is a trademark of Microware

☎ TOLL FREE
1-800-338-6800
SOUTH EAST MEDIA
5900 Cassandra Smith Rd.
Hixson, TN 37343
info (615) 842-4601
CoCo OS-9™ FLEX™
SOFTWARE

Availability Legends —
F = FLEX, CCF = Color Computer FLEX
O = OS-9, CCO = Color Computer OS-9
U = UniFLEX
CCD = Color Computer Disk
CCT = Color Computer Tape

!!! Please Specify Your Operating System & Disk Size !!!

FREE DISKETTE WITH EVERY $50 PURCHASE

☎ TOLL FREE
1-800-338-6800
For Ordering

TELEX 558 414 PVT BTH

SOUTH EAST MEDIA

5900 Cassandra Smith Rd.
Hixson, TN 37343
for information
call (615) 842-4601

CoCo OS-9™ FLEX™
SOFTWARE

DATA BASE MANAGEMENT SYSTEMS

Westchester Applied Business Systems

XDMS

Possibly one of the most powerful Database Management Systems' available. this machine language program is small enough to operate on a single sided 5" disk, yet provides the speed of M.L. and **power** limited only by the user's imagination. This DMS supports Relational, Sequential, Hierarchical, and Random Access File Structures, and has Virtual Memory capabilities for those Giant Data Bases. XDMS Level I provides a functional "entry level" System which provides for defining a Data Base, entering and changing the Data, and producing Reports. **XDMS Level II** adds the POWERFUL "GENERATE" facility which uses an English Language Command Structure in manipulating the Data to create new File Structures, Sort, Select, Calculate, etc. **XDMS Level III** adds several special "Utilities" which provide additional ease of working with the various structures, changing System Parameters, etc.

XDMS Lvl I - F & CCF - $129.95
XDMS Lvl II - F & CCF - $199.95
XDMS Lvl III - F & CCF - $269.95
XDMS System Manual only - $24.95

ACCOUNTING PACKAGES

Great Plains Computer Co. and **Universal Data Research, Inc.** both have Business Packages written in TSC XBASIC for FLEX, **CoCo FLEX**, and **UniFLEX** ----

• • • • • • • • • • • • •
- - - - Call **800-338-6800** for more information - - - -
• • • • • • • • • • • • •

Computer Systems Consultants

FULL SCREEN INVENTORY/MRP

The Full Screen Inventory System provides a means of maintaining small inventories. Using a linked, keyed random file structure based upon the item field, it keeps the file in alphabetical order for easier inquiry. With the FIND command, the user may locate and/or print all records matching on partial or complete item, description, vendor, or attributes. Items in backorder or below minimum stock levels may be located and/or printed thru the same process.. Printed output may be produced in item or vendor order. A materials requirement planning (MRP) capability for manufacturing environments is included to allow the maintenance and analysis of Hierarchical assemblies of items in the inventory file. It requires TSC's **Extended BASIC**.

F and CCF - **$100.00**, U - **$150.00**

The Virginia Company

BIZPACK

BIZPACK is used for storing accounting, numeric, and financial data which can then be used for planning, budgeting, forecasting, analyzing, etc. While "Electronic Spreadsheets" are extremely useful in many situations, BIZPACK excels in businesses where there are numerous expense columns, revenue sources, significant business indicators, large numbers, erratic week-to-week and month-to-month fluctuations, etc. BIZPACK helps determine statistical relationships, establish trend lines, "smooths" data via moving averages, analyze seasonal data, adjusts for inflation, lags data in Statistics or Column functions, plots data, etc. BIZPACK is oriented toward time series analysis of businesses. The Program displays information on the screen in Columns of Information with each Row conforming to a defined Period of Time (weeks, months, years, etc.), and is very easy to use (data is easy to enter, change, and modify; commands can be renamed to suit the users requirements; unlimited ability to create specialized commands using common BASIC Statements; etc.). Requires **TSC's Extended BASIC**.

F and CCF - **$135.00**
with Source - **$250.00**

Computer Systems Consultants

TABULA RASA SPREADSHEET

TABULA RASA is similar to DESKTOP/PLAN and provides for the generation and maintenance of tabular computation schemes often used for analysis of business, sales, and economic scenarios. Its menu-driven user interface provides these capabilities even to those users with no programming experience. Its extensive report-generation capabilities allow the user to generate professional results with minimum effort. It requires TSC's **Extended BASIC**.

F and CCF - **$100.00**, U - **$200.00**

Computer Systems Center

DYNACALC

THE Electronic Spread Sheet for 6809 Computer Systems. An extremely POWERFUL Business Tool, this Program will find an unlimited number of "non-business" applications, also (for example, a Full Junior College Electronics Curriculum was set up using DYNACALC). Advanced features like "Table Lookup" make Income Tax work easy; Column or Row Sorting for numerous applications; etc. Completely "Memory Resident", Machine Language, this Program is FAST. Provides STANDARD FLEX Text File output for use with BASIC, Word Processors, Pascal, "C", etc. Also available for Data-Comp and FHL FLEX systems using the 50 x 24 Displays.

F and **SPECIAL** CCF - **$200.00**
U - **$395.00**

ODDS AND ENDS

Computer Systems Consultants

FULL SCREEN FORMS DISPLAY

This package supports any Serial Terminal with cursor control of Memory-Mapped Video Displays. The package substantially extends the screen Input/Output capabilities of TSC's **Extended BASIC** programs by providing a simple, table-driven method of describing and using full screen displays. These table entries are easy to set up and maintain, and are normally stored on disk and read as required. A simple, interactive means of generating the forms and the data field definitions is provided.

F and CCF - **$50.00**, U - **$75.00**

Computer Systems Consultants

FULL SCREEN MAILING LIST

The Full Screen Mailing List System provides a means of maintaining simple mailing lists. Using a random fill structure based on the first character of the name field, it maintains the file in alphabetical order for easier inquiry. With the FIND command, the user may locate all records matching on partial or complete name, city, state, zip, or attributes. Printed listings and output to labels may also be produced on the same selective basis. It requires **TSC's Extended BASIC**.

F and CCF - **$100.00**, U - **$110.00**

VISA MasterCard

** SHIPPING **
Add 2% U.S.A.
(min. $2.50)
Add 5% Surface Foreign
10% Air Foreign

™FLEX is a trademark of Technical Systems Consultants
™OS9 is a trademark of Microware

☎ TOLL FREE
1-800-338-6800
SOUTH EAST MEDIA
5900 Cassandra Smith Rd.
Hixson, TN 37343
info (615) 842-4601
CoCo OS-9™ FLEX™
SOFTWARE

Availability Legends —
F = FLEX, CCF = Color Computer FLEX
O = OS-9, CCO = Color Computer OS-9
U = UniFLEX
CCD = Color Computer Disk
CCT = Color Computer Tape

!!! Please Specify Your Operating System & Disk Size !!!

Southeast Media

RAPIER
CHESS 6809

Requires FLEX and DISPLAYS On Any Type Terminal
Features:

*Four levels of play.
*Swap side. *Point scoring system.
*Two display boards. *Change skfll level.
*Solve Checkmate problems in 1-2-3-4 moves.
*Make move and swap sides. *Play white or black.

☆

This is one of the strongest CHESS programs running on any microcomputer, estimated USCF Rating 1600+ (better than most 'club' players at higher levels).

F and CCF - $79.95

Southeast Media

DIET-TRAC Forecaster

DIET-TRAC Forecaster is an XBASIC program that plans a diet in terms of either calories and percentage of carbohydrates, proteins and fats (C P 0%) or grams of Carbohydrate, Protein and Fat food exchanges of each of the six basic food groups (vegetable, bread, meat, skim milk, fruit and fat) for a specific individual.

Sex, Age, Height, Present Weight, Frame Size, Activity Level and Basal Metabolic Rate for normal individual are taken into account. Ideal weight and sustaining calories for any weight of the above individual are calculated. When a weight goal is given (either gain or loss), and a calorie plan is agreed upon between the computer and the individual, the number of days to reach the weight goal is projected. The starting and ending rate of weight loss is calculated, and a daily calendar with each day's weight for a 30-day period is printed.

F - $59.95
U - $89.95

COLOR COMPUTER SOFTWARE

Stearns Electronics

FORTH

Intrigued by Forth??? Here is a FORTH package tailored to the Color Computer! This package is supplied on Tape, with instructions for transferring it to **disk** if you wish. Written primarily in machine language, it's speed is unparalleled. A full Semigraphic-8 Editor is provided, along with "goodies" like Graphics and Sound Commands, Printer Commands, Auto-Repeat and Control Keys, etc. If you are interested in Learning FORTH, a Trace Feature is provided which is invaluable. If you are a FORTH Pro, this package provides CPU carry Flag accessibility, Fast Task Multiplexing, Clean Interrupt Handling, etc. (Or; you won't "out grow" the Basic capabilities of this Implementation). Combine this package with Leo Brodie's EXCELLENT Book "Starting FORTH", and you will be a FORTH Expert before you know it (and have a lot of fun doing it!).

Color Computer TAPE - $58.95

Custom Software Engineering, Inc.

Color Computer **GRAPHIC SCREEN PRINT** Programs

Dumps any "PMODE" Screen to the Printer with the BASIC USR Function. Shift the Printout Left or Right or Reverse Print (Dark for Light Screen and Vice Versa). All Programs on Tape.

GSPR for R.S. LP-VII/VIII & DMP 100/200/400	$7.95
GSPRE for Epson w/ Graftrax and Graftrax +	$9.95
GSPRG for Gemini 10 and 15	$9.95
GSPRP for the Prowriter Printers	$9.95

Custom Software Engineering, Inc.

DATE-O-BASE CALENDAR Program

A Menu Driven **EXTENDED BASIC** Program which allows the entry of up to 12 Memos per Day, each of which may contain up to 28 Characters, for any day of the Month between the years 1700 and 2099. A **Graphic Calendar** shows which days contain Memos, and a "Key Word" Search is provided which can be output to the Screen or Printer.

TAPE DATE-O-BASE CALENDAR
(Each Tape File will hold up to 400 Memos) $16.95
DISK DATE-O-BASE CALENDAR
(4,000 Memos at 300/Month per Disk) $19.95

FREE DISKETTE WITH EVERY $50 PURCHASE

☎ **TOLL FREE** **TELEX 558 414 PVT BTH**
1-800-338-6800
For Ordering

SOUTH EAST MEDIA

5900 Cassandra Smith Rd.
Hixson, TN 37343
for information
call (615) 842-4601

CoCo **OS-9™ FLEX™**
SOFTWARE

Custom Software Engineering, Inc.

That's INTEREST-ing

Interested in INTEREST (the Money Kind)? An EXTENDED BASIC Program that will help you deal with numerous problems requiring interest calculations. Present Value, Rate of Return, Current Bond Yield and Rate of Return to maturity, Loan Repayment Amortization Schedules, etc.

TAPE - $29.95

Custom Software Engineering, Inc.

DISK DATA HANDLER 64K

An EXTENDED BASIC Data Management System w/ Mach. Lang. Routines. Allows a max of 246 Chars. and 14 Ffelds per Record, and another Record can be linked to the first; 8 Char. Field Names, up to 99 Chars. per Field. Powerful On-Screen editor for input and update, flexible Output capabilities including output to Disk Files for use by other Programs. Change File Definition without re-entering the Data, Split Files, etc. Allows Multiple Field Sorts, Select on any combination of Fields, etc. An extremely POWERFUL TOOL; instructions provide examples of Mafling Lists and a Financial Stock Profit and Loss Tracking System.

DISK - $54.95

Custom Software Engineering, Inc.

DISK DOUBLE ENTRY

DISK EXTENDED BASIC Accounting Program w/ Mach. Lang. Routines. A "Traditional" Accounting Package for Small Business, Clubs, Churches, Personal Use, etc. Up to four levels of subtotals with Trial Balance, Income Statement, and Balance Sheet Reports. DDE allows up to 300 accounts and a Trial Balance of $9,999,999.99. Transactions may be up to 14 lines long, and comments and explanations may be freely used. Accounts are traceable to the Journal transaction, which may include comments. Screen reports allow review of past transactions and current balances.

DISK - $44.95

This Months ☆ STAR SPECIAL

'RAPIER' CHESS 6809
see page 39
Reg. $79.95

Special $39.95

Offer Ends April 15th, 1985

VISA MasterCard

**** SHIPPING ****
Add 2% U.S.A.
(min. $2.50)
Add 5% Surface Foreign
10% Air Foreign

™FLEX is a trademark of Technical Systems Consultants
™OS9 is a trademark of Microware

☎ TOLL FREE
1-800-338-6800
For Ordering
SOUTH EAST MEDIA
5900 Cassandra Smith Rd.
Hixson, TN 37343
Info (615) 842-4601
CoCo OS-9™ FLEX™
SOFTWARE

Availability Legends —
F = FLEX, **CCF** = Color Computer FLEX
O = OS-9, **CCO** = Color Computer OS-9
U = UniFLEX
CCD = Color Computer Disk
CCT = Color Computer Tape

!!! Please Specify Your Operating System & Disk Size !!!

```
                          *STRINGS:
C5D5 4A 61 6E             TABLE    FCC     'Jan'
C5D8 00                            FCB     0
C5D9 46 65 62                      FCC     'Feb'
C5DC 00                            FCB     0
C5DD 4D 61 72                      FCC     'Mar'
C5E0 00                            FCB     0
C5E1 41 70 72                      FCC     'Apr'
C5E4 00                            FCB     0
C5E5 4D 61 79                      FCC     'May'
C5E8 00                            FCB     0
C5E9 4A 75 6E                      FCC     'Jun'
C5EC 00                            FCB     0
C5ED 4A 75 6C                      FCC     'Jul'
C5F0 00                            FCB     0
C5F1 41 75 67                      FCC     'Aug'
C5F4 00                            FCB     0
C5F5 53 65 70                      FCC     'Sep'
C5F8 00                            FCB     0
C5F9 4F 63 74                      FCC     'Oct'
 5FC 00                            FCB     0
C5FD 4E 6F 76                      FCC     'Nov'
C600 00                            FCB     0
C601 44 65 63                      FCC     'Dec'
C604 00                            FCB     0

C605 43 61 74 61     STG1          FCC     'Catalog of Drive Number '
C609 6C 6F 67 20
C60D 6F 66 20 44
C611 72 69 76 65
C615 20 4E 75 6D
C619 62 65 72 20
C61D 04                            FCB     4
C61E 20 23 20 20     STG2          FCC     ' #    Name    Type  Size  '
C622 20 20 4E 61
C626 6D 65 20 20
C62A 20 20 54 79
C62E 70 65 20 20
C632 53 69 7A 65
C636 20 20
C638 20 20 44 61                   FCC     '  Date     Prot  Strt  End'
C63C 74 65 20 20
C640 20 20 20 50
C644 72 6F 74 20
C648 20 53 74 72
C64C 74 20 20 45
C650 6E 64
C652 0D 0A 04                      FCB     $D,$A,4
C655 44 69 73 6B     STG3          FCC     'Disk: '
C659 3A 20

C65B 04                            FCB     4
C65C 53 65 63 74     STG5          FCC     'Sectors Left = '
C660 6F 72 73 20
C664 4C 65 66 74
C668 20 3D 20
C66B 04                            FCB     4
C66C 53 59 4E 54     STG6          FCC     'SYNTAX ERROR, try again !!'
C670 41 58 20 45
C674 52 52 4F 52
C678 2C 20 74 72
C67  79 20 61 67
C680 61 69 6E 20
C684 21
C6 5 07 04                         F B     7.4

                C687 ZEND          EQU     *         END OF RAM FLAG

                                   END     CAT

0 ERROR(S) DETECTED
```

```
SYMBOL TABLE:

ADGK    C2CF   ADGK1   C2EE   AGNH    C2C3   AUTFG   C10E   CAT     C100
CAT0    C141   CAT01   C19F   CAT01A  C1B3   CAT01B  C1A4   CAT02   C1C4
CAT0D   C367   CAT2    C22D   CAT2A   C2     CAT2B   C290   CAT2Y   C19C
CAT2X   C287   CAT2Y   C289   CAT2Z   C26A   CAT4    C30D   CAT4A   C31C
CAT4B   C372   CAT4C   C32E   CAT6    C533   CAT6A   C539   CAT6B   C542
CAT9    C53C   CATFG   C114   CLMCHR  0043   CLMCNT  C108   CLMDFT  0000
CLMLIM  C115   DEFFG   C10F   DELCHR  0044   DELFG   C113   DRVFG   C112
ERRY    C297   ERRYX   C1C1   ERRZ    C29A   EXPCHR  0045   EXPFLG  C10C
EXT1    C3CE   FCB     C840   FILCNT  C110   FLEX    CD00   FMS     D406
FMSCLS  D403   FND0    C25E   FND1    C250   FND2    C25C   FND3    C25B
FOUND   C4A8   IDXTMP  C117   KEY     0044   KEY1    004F   KEY2    0047
LBPNTR  CC14   LP1     C416   LP1A    C41E   LP2     C42D   LP2A    C441
LP3     C450   LP4     C452   LP4A    C45A   LSTTRM  CC11   MAT0    C10F
MAT1    C1EC   MAT1A   C1EF   MAT3    C30E   MAT4    C390   MAT4A   C37D
MATFLG  C100   MTH     C4B4   MTH1    C4C8   NAMDE   C184   NCPRT   C339
NCPRT1  C36A   NNSP    C3F6   NSP     C487   NXTCH   CD27   NXTMNT  C498
OL01    C173   OLD1A   C175   OLDFG   C116   OPT1    C54E   OPT2    C551
OPT3    C569   OPT4    C572   OPT4A   C591   OPT5    C596   OPT5A   C581
OPT6    C588   OPTCHR  002B   OPTION  C54A   OUTADR  CD45   OUTDEC  CD39
PASS    0059   PCRLF   CD24   PMSG    CD1E   PMSG1   C5C7   PMSG2   C5D4
PROCHR  0050   PRT1    C4DF   PRT1A   C4DD   PRT2    C4EE   PRT2A   C4BC
PRT3    C4FD   PRT3A   C4FB   PRT4    C50C   PRT4A   C50A   PRT4B   C52D
PRT4D   C517   PSPACE  C5BE   PTRD    C109   PTRS    C107   PUTCHR  CD18
ROT     C1CE   RPTERR  CD3F   SAV1    C200   SAV2    C212   SAVPTR  C1FA
SHT1    C411   SIZE    C105   STEP    C49E   STG1    C605   STG2    C61E
STG3    C655   STG5    C65C   STG6    C66C   STGPTR  C119   STORE   C103
TABLE   C5D5   VN      C102   WARMS   CD03   WRKDRV  CC0C   ZEND    C687
```

ADDRESS

Here is a program that I call ADDRESS it is an address book program. The basic program is Robert Lund's RADCAT modified and changed to fit this application, so much of the credit should go to him. As R. Lund suggested in the July 83' issue of the Journal that the program might be useful as is or in another program, it has. Anyone using this program should refer to the July 83' Small DBMS articles for reference. My version uses two menu's, a main menu and one with the search command that allows you to change the Name, Address, City State & Zip, or Phone number. It allows you to delete an entry. The main allows you to print it on terminal or printer, exit to FLEX or insert a new entry. I hope this program is useful to someone. Again DON thanks for a GREAT Magazine.

Joseph M. Aulicino
2014-59th Street
Brooklyn, N.Y. 11204

```
                *       ADDRESS - J.M. Aulicino  10-Oct-83
                *
                * A MC6809 Address Book program
                * Menu Driven program
                *
                * Credit to R. Lund for his RADCAT program
                * from which I used many Routines...
                *
                *
                * Equates

      3FFF  MENTOP  EQU   $3FFF     - Top of table
      CD03  WARMS   EQU   $CD03     - Flex warm start
      CD1E  PSTRNG  EQU   $CD1E     - Flex print string
      CD24  PCRLF   EQU   $CD24     - Flex print CrLf
      CD15  GETCHR  EQU   $CD15     - Flex Get character
      CD18  PUTCHR  EQU   $CD18     - Flex Put character
      C840  FCB     EQU   $C840     - Flex File control block
      CD3F  RPTERR  EQU   $CD3F     - Flex Report error
      D403  FMSCLS  EQU   $D403     - FMS Close
      CD2D  GETFIL  EQU   $CD2D     - Flex Parse file specs
      CC14  BUFPNT  EQU   $CC14     - Flex Line buffer pointer
      D406  FMS     EQU   $D406     - Flex Call FMS
      E070  PRACIA  EQU   $E070     - Printer port
      E004  TMACIA  EQU   $E004     - Terminal port
      0FE0  PORT    EQU   $0FE0     - Flex I/O port in use
      CC09  PAUSE   EQU   $CC09     - Flex Pause flag

0100                ORG   $0100

                * Temp storage

0100                NEXTOP  RMB   2         - next entry
0102                TEMP1   RMB   2
0104                TEMP2   RMB   2
0106                TEMP3   RMB   2
0108                TEMP4   RMB   2
010A                TEMP5   RMB   2
010C                TSTRNG  RMB   22        - match string
```

```
                     * SET-UP TABLE
                     * zeros the table area

0122 8E   0778       TABCLR  LDX    #TABLE       - get table addr.
0125 4F                      CLRA
0126 A7   80         TACLR   STA    0,X+         - clear entry table
0128 8C   3FFF               CMPX   #MEMTOP
012B 26   F9                 BNE    TACLR

                     * initialize printer port

012D 86   03                 LDA    #$03         - reset ACIA
012F B7   E070               STA    PRACIA

0132 86   11                 LDA    #$11         - set ACIA to 8 bits 2 stop
0134 B7   E070               STA    PRACIA

                     * read file into memory
                     * starting at #TABLE

0137 BD   04A7               JSR    READ         - get address file

                     * FIND FIRST EMPTY ENTRY

013A 8E   0778       FIND    LDX    #TABLE       - point to table
013D A6   84                 LDA    0,X          - get 1st entry
013F 27   1A                 BEQ    FOUND        - if entry =0 found
0141 C6   5E         F1      LDB    #94          - entry size
0143 30   01         F2      INX                 - step thru table
0145 8C   3FFF               CMPX   #MEMTOP
0148 26   08                 BNE    NXT          - if not MEMTOP goto NXT
014A 8E   05AF               LDX    #MSG2        - Table Full
014D BD   CD1E               JSR    PSTRNG
0150 20   09                 BRA    FOUND
0152 5A              NXT     DECB                - dec & check if at next entry
0153 26   EE                 BNE    F2
0155 A6   84                 LDA    0,X
0157 27   02                 BEQ    FOUND
0159 20   E6                 BRA    F1
015B BF   0100       FOUND   STX    NEXTOP       - save when found

                     * DISPLAY OPTIONS & GET INPUT

015E BD   CD24       DISPLAY JSR    PCRLF
0161 8E   04FA               LDX    #MSG1        - display menu
0164 BD   CD1E               JSR    PSTRNG
0167 BD   CD15               JSR    GETCHR       - get char.
016A 81   31                 CMPA   #$31
016C 26   02                 BNE    DIS1         - is it a "1"
016E 20   1C                 BRA    ENTER
0170 81   32         DIS1    CMPA   #$32
0172 26   03                 BNE    DIS2         - is it a "2"
0174 7E   022C               JMP    SEARCH
0177 81   33         DIS2    CMPA   #$33
0179 26   03                 BNE    DIS3         - is it a "3"
017B 7E   0377               JMP    EXIT
017E 81   34         DIS3    CMPA   #$34
0180 26   02                 BNE    DIS4         - is it a "4"
0182 20   6D                 BRA    LIST
0184 81   35         DIS4    CMPA   #$35
0186 26   02                 BNE    DIS5         - is it a "5"
0188 20   7A                 BRA    PRINT
018A 20   D2         DIS5    BRA    DISPLAY      - if none of above do again

                     * ENTER INTO CATALOG

018C BD   CD24       ENTER   JSR    PCRLF
018F 8E   05CA               LDX    #MSG3        - Name
0192 BD   CD1E               JSR    PSTRNG
0195 BE   0100               LDX    NEXTOP       - pointer to next entry
0198 C6   16                 LDB    #22          - character count
019A 8D   33                 BSR    NAM1         - get name
019C BD   CD24               JSR    PCRLF
019F 8E   05D1               LDX    #MSG4        - Address
01A2 BD   CD1E               JSR    PSTRNG
01A5 BE   0100               LDX    NEXTOP       - pointer to next entry
01A8 C6   1E                 LDB    #30          - character count
01AA 8D   23                 BSR    NAM1         - get address
01AC BD   CD24               JSR    PCRLF
01AF 8E   05DB               LDX    #MSG5        - City, State
01B2 BD   CD1E               JSR    PSTRNG
```

```
01B5 BE   0100               LDX    NEXTOP       - pointer to next entry
01B8 C6   1E                 LDB    #30          - character count
01BA 8D   13                 BSR    NAM1         - get city, state & zip
01BC BD   CD24               JSR    PCRLF
01BF 8E   05F2               LDX    #MSG6        - Phone#
01C2 BD   CD1E               JSR    PSTRNG
01C5 BE   0100               LDX    NEXTOP       - Pointer to next entry
01C8 C6   0C                 LDB    #12          - character count
01CA 8D   03                 BSR    NAM1         - get phone #
01CC 7E   013A               JMP    FIND

                     * LOAD FIELDS

01CF BD   CD15       NAM1    JSR    GETCHR       - get char.
01D2 81   0D                 CMPA   #$0D
01D4 27   10                 BEQ    NAM2         - if Cr end entry
01D6 81   08                 CMPA   #$08
01D8 26   05                 BNE    NAMB         - if not BS enter char.
01DA 30   1F                 DEX                 - back up pointer
01DC 5C                      INCB                - add to char. count
01DD 20   F0                 BRA    NAM1         - get next char.
01DF A7   80         NAMB    STA    0,X+         - store char. in table
01E1 5A                      DECB                - decrement char. count
01E2 26   EB                 BNE    NAM1         - if not 0 get next char.
01E4 20   07                 BRA    NAM3         - end char. entry
01E6 86   20         NAM2    LDA    #$20         - add spaces till char. count =0
01E8 A7   80                 STA    0,X+
01EA 5A                      DECB
01EB 26   F9                 BNE    NAM2
01ED BF   0100       NAM3    STX    NEXTOP
01F0 39                      RTS

                     * LIST ENTIRE BOOK

01F1 8E   0778       LIST    LDX    #TABLE       - point to beginning of table
01F4 BF   0100               STX    NEXTOP
01F7 BD   0382       LIST3   JSR    OUT          - output entry
01FA BE   0100               LDX    NEXTOP
01FD 27   02                 BEQ    LIST4        - if NEXTOP is zero end
01FF 20   F6                 BRA    LIST3
0201 7E   013A       LIST4   JMP    FIND

                     * LIST ON PRINTER

0204 8E   E070       PRINT   LDX    #PRACIA
0207 BF   DFE0               STX    PORT         - change I/O port to printer/
020A 8E   0778               LDX    #TABLE       - point to top of table
020D BF   0100               STX    NEXTOP
0210 86   FF                 LDA    #$FF
0212 B7   CC09               STA    PAUSE        - disable pause
0215 BD   0382       PRINT1  JSR    OUT          - print entry
0218 BE   0100               LDX    NEXTOP
021B 27   02                 BEQ    PRINT2       - if NEXTOP is zero end
021D 20   F6                 BRA    PRINT1
021F 8E   E004       PRINT2  LDX    #TMACIA
0222 BF   DFE0               STX    PORT         - restore terminal port
0225 4F                      CLRA
0226 B7   CC09               STA    PAUSE        - restore pause
0229 7E   013A               JMP    FIND

                     * SEARCH FOR NAME

022C BD   CD24       SEARCH  JSR    PCRLF
022F 8E   063B               LDX    #MSG11       - Search for
0232 BD   CD1E               JSR    PSTRNG
0235 8E   010C               LDX    #TSTRNG      - point to test string
0238 BD   CD15       SEA1    JSR    GETCHR
023B 81   0D                 CMPA   #$0D         - if Cr end string
023D 27   11                 BEQ    SEA2
023F 81   08                 CMPA   #$08
0241 26   04                 BNE    SEA10        - if not BS store char.
0243 30   1F                 DEX                 - decrement pointer
0245 20   F1                 BRA    SEA1         - get next char.
0247 A7   80         SEA10   STA    0,X+         - store char. in test string
0249 8C   0122               CMPX   #TSTRNG+22
024C 27   02                 BEQ    SEA2
024E 20   E8                 BRA    SEA1         - get next char.
0250 4F              SEA2    CLRA                - end string
0251 A7   84                 STA    0,X
0253 BD   CD24               JSR    PCRLF
0256 8E   071A               LDX    #TABLE-94
```

```
0259 BF   0108              STX   TEMP4     - save pointer
025C 86   16        SEA3    LDA   #22
025E B7   0106              STA   TEMP3     - store char. count
0261 BE   0108              LDX   TEMP4
0264 C6   5E                LDB   #94       - entry size
0266 30   01        SEA4    INX
0268 8C   3FFF              CMPX  #MEMTOP
026B 26   02                BNE   SEA5      - if not MEMTOP
026D 20   6B                BRA   FIN
026F 5A             SEA5    DECB            - decrement entry count
0270 26   F4                BNE   SEA4
0272 BF   0108              STX   TEMP4     - save current entry
0275 BF   0104              STX   TEMP2     - save table pointer
0278 8E   010C      SEA9    LDX   #TSTRNG   - point to test string

027B BF   0102              STX   TEMP1

                    * Get 'B' from TSTRNG

027E BE   0102      SEA6    LDX   TEMP1     - get pointer
0281 E6   84                LDB   0,X       - get char.
0283 26   02                BNE   SEA7      - Done ?
0285 20   1F                BRA   MATCH     - all matched
0287 30   01        SEA7    INX             - bump pointer
0289 BF   0102              STX   TEMP1     - save it

                    * get 'A' from table/

028C BE   0104      SEA8    LDX   TEMP2     - get table pointer
028F A6   80                LDA   0,X+      - get char.
0291 8C   3FFF              CMPX  #MEMTOP   - done ?
0294 27   44                BEQ   FIN       - if yes exit
0296 7A   0106              DEC   TEMP3     - decrement char. count
0299 27   C1                BEQ   SEA3      - if count =0 go next entry
029B BF   0104              STX   TEMP2
029E 34   04 A1E0           CBA             - compare char.
02A2 27   DA                BEQ   SEA6      - if match get next char.
02A4 20   D2                BRA   SEA9      - reposition pointer
02A6 BE   0108      MATCH   LDX   TEMP4     - point to entry
02A9 BF   0100              STX   NEXTOP
02AC BF   010A              STX   TEMP5
02AF A6   84                LDA   0,X
02B1 27   A9                BEQ   SEA3      - if open entry find next entry
02B3 BD   039C              JSR   OUT3      - print it
02B6 8E   065F              LDX   #MSG13    - Change this entry ?
02B9 BD   CD1E              JSR   PSTRNG
02BC BD   CD15              JSR   GETCHR
02BF 81   59                CMPA  #$59
02C1 26   99                BNE   SEA3      - if no get next entry
02C3 8E   05FC              LDX   #MSG7     - Are you sure ?
02C6 BD   CD1E              JSR   PSTRNG
02C9 BD   CD15              JSR   GETCHR
02CC 81   59                CMPA  #$59
02CE 26   8C                BNE   SEA3      - if no get next entry
02D0 20   18                BRA   CHANGE    - goto change routine
02D2 BE   010A      DELETE  LDX   TEMP5     - get entry pointer
02D5 4F                     CLRA
02D6 A7   84                STA   0,X       - clear first byte of entry
02D8 20   82                BRA   SEA3
02DA BD   CD24      FIN     JSR   PCRLF
02DD 8E   0648              LDX   #MSG12    - (CR) to continue
02E0 BD   CD1E              JSR   PSTRNG
02E3 BD   CD15              JSR   GETCHR    - wait for keystroke
02E6 4F                     CLRA
02E7 7E   013A              JMP   FIND

                    * CHANGE ENTRY

02EA 8E   0674      CHANGE  LDX   #MSG14

02ED BD   CD1E              JSR   PSTRNG
02F0 8E   0705              LDX   #MSG15    - print menu
02F3 BD   CD1E              JSR   PSTRNG
02F6 BD   CD15              JSR   GETCHR
02F9 81   31                CMPA  #$31
02FB 27   18                BEQ   NAME      - if '1' goto name
02FD 81   32                CMPA  #$32
02FF 27   27                BEQ   ADDR      - if '2' goto addr
0301 81   33                CMPA  #$33
0303 27   3D                BEQ   CITY      - if '3' goto city
0305 81   34                CMPA  #$34
0307 27   53                BEQ   PHONE     - if '4' goto phone
0309 81   35                CMPA  #$35
030B 27   C5                BEQ   DELETE    - if '5' goto delete
030D 81   36                CMPA  #$36
030F 1027 FE27              LBEQ  FIND      - if '6' goto other menu
0313 20   D5                BRA   CHANGE
0315 BD   CD24      NAME    JSR   PCRLF
0318 8E   05CA              LDX   #MSG3     - Name
031B BD   CD1E              JSR   PSTRNG
031E BE   010A              LDX   TEMP5     - get pointer
0321 C6   16                LDB   #22       - set char. count
0323 BD   01CF              JSR   NAM1      - get entry
0326 20   C2                BRA   CHANGE    - goto change menu
0328 BD   CD24      ADDR    JSR   PCRLF
032B 8E   05D1              LDX   #MSG4     - Address
032E BD   CD1E              JSR   PSTRNG
0331 BE   010A              LDX   TEMP5     - get pointer
0334 C6   16                LDB   #22       - get name count
0336 30   01        ADDR1   INX             - skip over name entry
0338 5A                     DECB
0339 26   FB                BNE   ADDR1
033B C6   1E                LDB   #30       - char. count
033D BD   01CF              JSR   NAM1      - get entry
0340 20   A8                BRA   CHANGE    - goto change menu
0342 BD   CD24      CITY    JSR   PCRLF
0345 8E   05DB              LDX   #MSG5     - City, State & zip
0348 BD   CD1E              JSR   PSTRNG
034B BE   010A              LDX   TEMP5     - get pointer
034E C6   34                LDB   #52       - get name + address count
0350 30   01        CITY1   INX             - skip over name + address entry
0352 5A                     DECB
0353 26   FB                BNE   CITY1
0355 C6   1E                LDB   #30       - set char. count
0357 BD   01CF              JSR   NAM1      - get entry
035A 20   8E                BRA   CHANGE    - goto change menu
035C BD   CD24      PHONE   JSR   PCRLF
035F 8E   05F2              LDX   #MSG6     - Phone#
0362 BD   CD1E              JSR   PSTRNG
0365 BE   010A              LDX   TEMP5     - get pointer
0368 C6   52                LDB   #82       - get name + address + city count
036A 30   01        PHONE1  INX             - skip name, addr. & city entry
036C 5A                     DECB
036D 26   FB                BNE   PHONE1

036F C6   0C                LDB   #12       - set char. count
0371 BD   01CF              JSR   NAM1      - get entry
0374 16   FF73              LBRA  CHANGE

                    * EXIT ROUTINE

0377 BD   047C      EXIT    JSR   DELFIL    - delete old .BAK file
037A 8D   7E                BSR   RENAME    - renames .DAT to .BAK
037C BD   042C              JSR   WRITE     - writes new .DAT
037F 7E   CD03              JMP   WARMS     - exit to FLEX

                    * OUT

0382 BE   0100      OUT     LDX   NEXTOP    - point to entry
0385 A6   84                LDA   0,X       - get 1st char.
0387 26   13                BNE   OUT3      - if not 0 print it
0389 C6   5E                LDB   #94       - get next entry
038B 30   01        OUT1    INX
038D 8C   3FFF              CMPX  #MEMTOP
0390 26   02                BNE   OUT2
0392 20   5F                BRA   OUT8      - exit
0394 5A             OUT2    DECB
0395 26   F4                BNE   OUT1
0397 BF   0100              STX   NEXTOP
039A 20   E6                BRA   OUT
039C BD   CD24      OUT3    JSR   PCRLF     - out CrLf
039F C6   16                LDB   #22       - set char. count
03A1 A6   84        OUT4    LDA   0,X       - get char.
03A3 BD   CD18              JSR   PUTCHR    - output char.
03A6 30   01                INX
03A8 8C   3FFF              CMPX  #MEMTOP
03AB 27   46                BEQ   OUT8      - if at MEMTOP exit
03AD 5A                     DECB            - decrement char. count
03AE 26   F1                BNE   OUT4      - if not 0 get next char.
03B0 BD   CD24              JSR   PCRLF
03B3 C6   1E                LDB   #30       - set char. count
```

```
03B5 A6    84      OUT5   LDA   0,X      - get char.
03B7 BD    CD18           JSR   PUTCHR   - output char.
03BA 30    01             INX
03BC 8C    3FFF           CMPX  #MEMTOP
03BF 27    32             BEQ   OUT8     - if at MEMTOP exit
03C1 5A                   DECB           - decrement char. count
03C2 26    F1             BNE   OUT5     - if not 0 get next char.
03C4 BD    CD24           JSR   PCRLF
03C7 C6    1E             LDB   #30      - set char. count
03C9 A6    84      OUT6   LDA   0,X      - get char.
03CB BD    CD18           JSR   PUTCHR   - output char.
03CE 30    01             INX
03D0 8C    3FFF           CMPX  #MEMTOP
03D3 27    1E             BEQ   OUT8     - if at MEMTOP exit
03D5 5A                   DECB           - decrement char. count
03D6 26    F1             BNE   OUT6     - if not 0 get next char.
03D8 BD    CD24           JSR   PCRLF
03DB C6    0C             LDB   #12      - set char. count

03DD A6    84      OUT7   LDA   0,X      - output char.
03DF BD    CD18           JSR   PUTCHR
03E2 30    01             INX
03E4 8C    3FFF           CMPX  #MEMTOP
03E7 27    0A             BEQ   OUT8     - if at MEMTOP exit
03E9 5A                   DECB           - decrement char. count
03EA 26    F1             BNE   OUT7     - if not 0 get next char.
03EC BD    CD24           JSR   PCRLF
03EF BF    0100           STX   NEXTOP   - store current pointer
03F2 39                   RTS
03F3 7F    0100    OUT8   CLR   NEXTOP
03F6 7F    0101           CLR   NEXTOP+1
03F9 39                   RTS

                   * RENAME FILE ROUTINE

03FA 8E    C875    RENAME LDX   #FCB+53  - put .BAK name into FCB
03FD 108E  0622           LDY   #MSG9
0401 A6    A0      REN1   LDA   0,Y+     - get name char.
0403 81    04             CMPA  #4
0405 27    04             BEQ   REN2     - check for end of name
0407 A7    80             STA   0,X+     - store char. in FCB
0409 20    F6             BRA   REN1     - get next char.
040B 8E    0615    REN2   LDX   #MSG8
040E BF    CC14           STX   BUFPNT   - set pointer to .DAT name
0411 8E    C840           LDX   #FCB
0414 BD    CD2D           JSR   GETFIL   - parse file spec
0417 25    0A             BCS   ERROR4
0419 86    0D             LDA   #13      - rename function code
041B A7    84             STA   0,X
041D BD    D406           JSR   FMS      - do rename function
0420 26    01             BNE   ERROR4
0422 39                   RTS
0423 BD    CD3F    ERROR4 JSR   RPTERR   - report error
0426 BD    D403           JSR   FMSCLS   - close file
0429 7E    CD03           JMP   WARMS    - exit to Flex

                   * WRITE TO DISK ROUTINE

042C 8E    0778    WRITE  LDX   #TABLE
042F BF    0102           STX   TEMP1    - save Pointer
0432 8E    0615           LDX   #MSG8
0435 BF    CC14           STX   BUFPNT   - set pointer to file name
0438 8E    C840           LDX   #FCB
043B BD    CD2D           JSR   GETFIL   - parse file spec
043E 25    33             BCS   WRITE3
0440 86    02             LDA   #2       - write function code
0442 A7    84             STA   0,X
0444 BD    D406           JSR   FMS      - do open for write
0447 26    2A             BNE   WRITE3
0449 6F    84             CLR   0,X
044B 86    FF             LDA   #$FF
044D A7    88 3B          STA   59,X     - no space compression

                   * WRITE MEMORY IMAGE TO DISK

0450 8E    C840    WRITE1 LDX   #FCB     - FCB pointer
0453 10BE  0102           LDY   TEMP1    - table pointer
0457 A6    A0      WRITE4 LDA   0,Y+     - get char. from table
0459 108C  3FFF           CMPY  #MEMTOP
045D 27    07             BEQ   WRITE2   - if at MEMTOP close file
045F BD    D406           JSR   FMS      - write char. to file
```

```
0462 26    0F             BNE   WRITE3
0464 20    F1             BRA   WRITE4

                   * CLOSE FILE & EXIT

0466 8E    C840    WRITE2 LDX   #FCB     - point to FCB
0469 86    04             LDA   #4       - close function
046B A7    84             STA   0,X
046D BD    D406           JSR   FMS      - do close
0470 26    01             BNE   WRITE3
0472 39                   RTS

                   * ERROR HANDLER

0473 BD    CD3F    WRITE3 JSR   RPTERR   - report error
0476 BD    D403           JSR   FMSCLS   - close file
0479 7E    CD03           JMP   WARMS    - exit to Flex

                   * DELETE FILE

047C 8E    062E    DELFIL LDX   #MSG10
047F BF    CC14           STX   BUFPNT   - set pointer to file name
0482 8E    C840           LDX   #FCB     - point to FCB
0485 BD    CD2D           JSR   GETFIL
0488 25    0A             BCS   DEL1     - if error close file
048A 86    0C             LDA   #12      - delete function code
048C A7    84             STA   0,X
048E BD    D406           JSR   FMS      - do delete function
0491 26    01             BNE   DEL1     - if error close file
0493 39                   RTS
0494 8E    C840    DEL1   LDX   #FCB     - point to FCB
0497 A6    01             LDA   1,X
0499 81    04             CMPA  #4
049B 26    01             BNE   DEL2     - if not EOF error close
049D 39                   RTS
049E BD    CD3F    DEL2   JSR   RPTERR   - report error
04A1 BD    D403           JSR   FMSCLS   - close file
04A4 7E    CD03           JMP   WARMS    - exit to Flex

                   * READ FILE

04A7 8E    0615    READ   LDX   #MSG8
04AA BF    CC14           STX   BUFPNT   - set pointer to file name
04AD 8E    C840           LDX   #FCB     - point to FCB
04B0 BD    CD2D           JSR   GETFIL   - parse file spec
04B3 25    39             BCS   READ3

04B5 86    01             LDA   #1       - read function code
04B7 A7    84             STA   0,X
04B9 BD    D406           JSR   FMS      - do read function
04BC 26    30             BNE   READ3
04BE 6F    84             CLR   0,X
04C0 86    FF             LDA   #$FF     - set for no space compression
04C2 A7    88 3B          STA   59,X

                   * READ MEMORY IMAGE

04C5 8E    C840    READ1  LDX   #FCB     - point to FCB
04C8 108E  0778           LDY   #TABLE   - point to table
04CC BD    D406    READ5  JSR   FMS      - get char. from file
04CF 26    0A             BNE   READ2
04D1 A7    A0             STA   0,Y+     - store in table
04D3 108C  3FFF           CMPY  #MEMTOP
04D7 27    0B             BEQ   READ4    - if at MEMTOP go close file
04D9 20    F1             BRA   READ5

                   * END OF FILE ERROR

04DB 8E    C840    READ2  LDX   #FCB
04DE A6    01             LDA   1,X      - get error code
04E0 81    08             CMPA  #8
04E2 26    0A             BNE   READ3    - if not EOF error go other error

                   * CLOSE FILE & EXIT

04E4 86    04      READ4  LDA   #4       - close file function
04E6 A7    84             STA   0,X
04E8 BD    D406           JSR   FMS      - do file close
04EB 26    01             BNE   READ3
04ED 39                   RTS
```

```
                    * NO FILE ERROR

04EE BE   CB40      READ3  LDX    #FCB
04F1 A6   01               LDA    1,X          - get error code
04F3 81   04               CMPA   #4
04F5 26   E9               BNE    READ4        - if not no file error go close
04F7 7E   042C             JMP    WRITE

                    * Messages

04FA 20 20 2A 2A    MSG1   FCC    '  ***** Phone Book *****',$D,$A
04FE 2A 2A 2A 20
0502 50 68 6F 6E
0506 65 20 42 6F
050A 6F 6B 20 2A
050E 2A 2A 2A 2A
0512 0D 0A
0514 4E 65 77 20           FCC    'New Entry.............(1)',$D,$A
0518 45 6E 74 72
051C 79 2E 2E 2E
0520 2E 2E 2E 2E
0524 2E 2E 2E 2E
0528 2E 2E 28 31
052C 29 0D 0A
052F 53 65 61 72           FCC    'Search for name.......(2)',$D,$A
0533 63 68 20 66
0537 6F 72 20 6E
053B 61 6D 65 2E
053F 2E 2E 2E 2E
0543 2E 2E 28 32
0547 29 0D 0A
054A 45 78 69 74           FCC    'Exit to Flex..........(3)',$D,$A
054E 20 74 6F 20
0552 46 6C 65 78
0556 2E 2E 2E 2E
055A 2E 2E 2E 2E
055E 2E 2E 28 33
0562 29 0D 0A
0565 4C 69 73 74           FCC    'List  book............(4)',$D,$A
0569 20 20 62 6F
056D 6F 6B 2E 2E
0571 2E 2E 2E 2E
0575 2E 2E 2E 2E
0579 2E 2E 28 34
057D 29 0D 0A
0580 50 72 69 6E           FCC    'Print Book............(5)',$D,$A,$A,$A
0584 74 20 42 6F
0588 6F 6B 2E 2E
058C 2E 2E 2E 2E
0590 2E 2E 2E 2E
0594 2E 2E 28 35
0598 29 0D 0A 0A
059C 0A
059D 45 6E 74 65           FCC    'Enter Selection: ',4
05A1 72 20 53 65
05A5 6C 65 63 74
05A9 69 6F 6E 3A
05AD 20 04
05AF 2A 2A 2A 20    MSG2   FCC    '*** ALL ENTRIES FULL ***',$D,$A,4
05B3 41 4C 4C 20
05B7 45 4E 54 52
05BB 49 45 53 20
05BF 46 55 4C 4C
05C3 20 2A 2A 2A
05C7 0D 0A 04
05CA 4E 61 6D 65    MSG3   FCC    'Name: ',4
05CE 3A 20 04
05D1 41 64 64 72    MSG4   FCC    'Address: ',4
05D5 65 73 73 3A
05D9 20 04
05DB 43 69 74 79    MSG5   FCC    'City, State Zip code: ',4
05DF 2C 20 53 74
05E3 61 74 65 20
05E7 5A 69 70 20
05EB 63 6F 64 65
05EF 3A 20 04
05F2 50 68 6F 6E    MSG6   FCC    'Phone #: ',4
05F6 65 20 23 3A
05FA 20 04
05FC 41 72 65 20    MSG7   FCC    'Are you sure? (Y or N): ',4
0600 79 6F 75 20
0604 73 75 72 65
0608 3F 20 28 59
060C 20 6F 72 20
0610 4E 29 3A 20
0614 04
0615 41 44 44 52    MSG8   FCC    'ADDRBOOK.DAT',$D
0619 42 4F 4F 4B
061D 2E 44 41 54
0621 0D
0622 41 44 44 52    MSG9   FCC    'ADDRBOOKBAK',4
0626 42 4F 4F 4B
062A 42 41 4B 04
062E 41 44 44 52    MSG10  FCC    'ADDRBOOK.BAK',$D
0632 42 4F 4F 4B
0636 2E 42 41 4B
063A 0D
063B 53 65 61 72    MSG11  FCC    'Search for: ',4
063F 63 68 20 66
0643 6F 72 3A 20
0647 04
0648 45 6E 74 65    MSG12  FCC    'Enter <CR> to continue',4
064C 72 20 3C 43
0650 52 3E 20 74
0654 6F 20 63 6F
0658 6E 74 69 6E
065C 75 65 04
065F 43 68 61 6E    MSG13  FCC    'Change this entry ? ',4
0663 67 65 20 74
0667 68 69 73 20
066B 65 6E 74 72
066F 79 20 3F 20
0673 04
0674 4E 61 6D 65    MSG14  FCC    'Name...............(1)',$D,$A
0678 2E 2E 2E 2E
067C 2E 2E 2E 2E
0680 2E 2E 2E 2E
0684 2E 2E 2E 28
0688 31 29 0D 0A
068C 41 64 64 72           FCC    'Address............(2)',$D,$A
0690 65 73 73 2E
0694 2E 2E 2E 2E
0698 2E 2E 2E 2E
069C 2E 2E 2E 28
06A0 32 29 0D 0A
06A4 43 69 74 79           FCC    'City, State........(3)',$D,$A
06A8 2C 20 53 74
06AC 61 74 65 2E
06B0 2E 2E 2E 2E
06B4 2E 2E 2E 28
06B8 33 29 0D 0A
06BC 50 68 6F 6E           FCC    'Phone #............(4)',$D,$A
06C0 65 20 23 2E
06C4 2E 2E 2E 2E
06C8 2E 2E 2E 2E
06CC 2E 2E 2E 28
06D0 34 29 0D 0A
06D4 44 65 6C 65           FCC    'Delete entry.......(5)',$D,$A
06D8 74 65 20 65
06DC 6E 74 72 79
06E0 2E 2E 2E 2E
06E4 2E 2E 2E 28
06E8 35 29 0D 0A
06EC 4E 6F 20 63           FCC    'No change..........(6)',$D,$A,4
06F0 68 61 6E 67
06F4 65 2E 2E 2E
06F8 2E 2E 2E 2E
06FC 2E 2E 2E 28
0700 36 29 0D 0A
0704 04
0705 43 68 61 6E    MSG15  FCC    'Change which: ',4
0709 67 65 20 77
070D 68 69 63 68
0711 3A 20 04

               0778 TABLE  EQU    *+100
                           END    TABCLR

0 ERROR(S) DETECTED
```

```
SYMBOL TABLE:

ADDR   0328   ADDR1  0336   BUFPNT CC14   CHANGE 02EA   CITY   0342
CITYL  0350   DEL1   0494   DEL2   049E   DELETE 02B2   DELFIL 047C
DIS1   0170   DIS2   0177   DIS3   017E   DIS4   01B4   DIS5   01BA
DISPLA 015E   ENTER  018C   ERROR4 0423   EXIT   0377   F1     0141
F2     0143   FCB    CB40   FIN    020A   FIND   013A   FMS    0406
FMSCLS D403   FOUND  0158   GETCHR CD15   GETFIL CD2D   LIST   01F1
LIST3  01F7   LIST4  0201   MATCH  0246   MEMTOP 3FFF   MSG1   04FA
MSG10  062E   MSG11  063B   MSG12  0648   MSG13  065F   MSG14  0674
MSG15  0705   MSG2   05AF   MSG3   05CA   MSG4   05D1   MSG5   05D8
MSG6   05F2   MSG7   05FC   MSG8   0615   MSG9   0622   NAM1   01CF
NAM2   01E6   NAM3   01ED   NAMB   01DF   NAME   0315   NEXTOP 0100
NXT    0152   OUT    0382   OUT1   0388   OUT2   0394   OUT3   039C
OUT4   03A1   OUT5   03B5   OUT6   03C9   OUT7   03DD   OUT8   03F3
PAUSE  CC09   PCRLF  CD24   PHONE  035C   PHONE1 036A   PORT   BFE0
PRACIA E070   PRINT  0204   PRINT1 0215   PRINT2 021F   PSTRNG CD1E
PUTCHR CD18   READ   04A7   READ1  04C5   READ2  04DB   READ3  04EE
READ4  04E4   READS  04CC   REN1   0401   REN2   040B   RENAME 03FA
RPTERR CD3F   SEA1   0238   SEA10  0247   SEA2   0250   SEA3   025C
SEA4   0266   SEA5   026F   SEA6   027E   SEA7   0287   SEA8   028C
SEA9   027B   SEARCH 022C   TABCLR 0122   TABLE  0278   TACLR  0126

TEMP1  0102   TEMP2  0104   TEMP3  0106   TEMP4  0108   TEMP5  010A
TRACIA E004   TSTRNG 010C   WARMS  CD03   WRITE  042C   WRITE1 0450
WRITE2 0466   WRITE3 0473   WRITE4 0457
```

ANALOG TO DIGITAL, DIGITAL TO ANALOG

12 BIT ANALOG to DIGITAL, DIGITAL to ANALOG CONVERTER

Ron Anderson has designed a 12 bit 16 channel Analog to Digital converter (to which I have made some modifications) using a National Semiconductor ADC1210HCD; and I have assembled a 12 bit Digital to Analog converter using an Analog Devices AD567KD to be constructed on a Thomas Instrumentation SP-1 board. Tom Gluyas of Thomas Instrumentation has agreed to sell the SP-1 without the 4 6850's for $175 assembled and tested. If the analog amplifiers are built on the board, there will not be sufficient room to be able to utilize the 6850's unless no other parts are needed. (I assembled the bare board and had spent about $125 to get the board ready for wire wrapping. I wish that I had asked about buying the board partially populated--the extra $50 would have been well worth the cost.) The parts cost for the basic AD and DA circuits is about $175 with each 0-20 mv amplifier being an additional $30.

D/A CONVERTER

The D/A converter is designed to give a +/- 5 volt output range for a digital input of 4095 and 0000 respectively (FFF to 000 hexadecimal). Actually the output is adjusted to +4.9976 and -5.0000 for these inputs as most calibration procedures recommend adjusting the output to 1 LSB less than the nominal range. (The device may be readily reconfigured to give 0 to +10, +/- 10 or +/- 2.5 volt output. A different output amplifier might be necessary in order to implement the 10 volt versions as the 12 volt supplies do not usually have sufficient drive to make a bipolar transistor output stage go to 10 volts.) The circuitry is based on that in the Analog Devices data sheet for the AD567, but the 6821 allows the device to be wired as if the computer had a 12 line data bus with a 2 line control bus. The diagram for the converter and the 5.000 volt reference is given on Figure 1.

Through use of the 6821 PIA, the computer may be made to appear as if it had a 12 line data bus. The 6821 is set up to simultaneously output a 12 bit word to the AD567 with the 8 LSB's stored in the side A registers and the 4 MSB's stored in the side B ones. When the MSB's are read into the 6821, bit 7 must also be forced to 0 as it is used to control the NOT(WRITE) input of the D/A. The CB2 output of the 6821 is programmed to momentarily go low on the next E clock transition after the MSB's have been strobed into the AD567. The chip latches the data and then makes the conversion which may take 500 ns during which time the 6821 may be readdressed. The procedure in the sample software should be followed in order to properly set up the 6821 and to strobe the data into the AD567.

To calibrate the AD567, first send $0000 to the converter and adjust trimmer R1 to give exactly -5.000 volts. Then send $0FFF to the chip and adjust R2 to give +4.9976 volts. Recheck both adjustments and the calibration is complete.

If you want to have some fun comparing the speed of assembler, PL9 and TSC BASIC programs, then try the following ones which generate a sawtooth. They also allow you to see if the hardware is functioning properly. Be prepared, however, the assembler and PL9 programs will need an oscilloscope to monitor the output giving values of about 43 and 75 millisec respectively, whereas the BASIC program will require a strip chart recorder or a very slow sweep on the scope as it takes 13.5 seconds!

```
*       SAWTOOTH WAVEFORM GENERATOR PROGRAM FOR AD567K ON
A
*       THOMAS INSTRUMENTATION SP-1 BOARD
*       BOARD BASE ADDRESS AT $E100 AS PER GIMIX FLEX
*       PIA 11 ON BOARD USED FOR D/A
**************************************************
*       by J. A. McDaniel                         *
*       University of Maine at Farmington          *
*       39 High Street                            *
*       Farmington, Maine 04938                    *
*       August 15, 1984                           *
**************************************************
*

START EQU $0000
      LDA #$0
      STA $E111   BASE ADDRESS OF SIDE A CONTROL REGISTER
*               PREPARE TO ADDRESS DATA DIRECTION A
      STA $E113   BASE ADDRESS OF SIDE B CONTROL REGISTER
*               PREPARE TO ADDRESS DATA DIRECTION B
      LDA #$FF
      STA $E110   SIDE A DATA DIRECTION, SET FOR OUTPUT
      STA $E112   SIDE B DATA DIRECTION, SET FOR OUTPUT
      LDA #$04
      STA $E111   SELECT SIDE A OUTPUT REGISTER, NO
INTERRUPTS
      LDA #$2C
      STA $E113   SELECT SIDE B OUTPUT REGISTER, AND
*               SET CB2 AS OUTPUT NO INTERRUPTS
      CLRA
      CLRB
AGAIN LDD #$0000
UP    ADDD #$0001
      STB $E110   LOAD LOW ORDER BITS
      STA $E112   LOAD HIGH ORDER BITS AND FORCE
NOT(WRITE) LOW
      CMPD #$0FFF
      BNE UP
      BRA AGAIN
      END START
```

```
/* D/A CONVERTER TEST: SEPT 14, 1984   GIVES SAWTOOTH */
/* THOMAS INSTRUMENTATION SP-1 BOARD ADDRESSED AT $E100
AS PER
GIMIX FLEX, D/A USES PIA 11 AT $E110 TO $E113 */

/* BY J. A. MCDANIEL
   UNIVERSITY OF MAINE AT FARMINGTON
   39 HIGH STREET
   FARMINGTON, MAINE 04938 */

ORIGIN = $0000
GLOBAL BYTE LBIT, HBIT;
STACK *;

AT $E110: BYTE  DDL(0),  DATAL,  CONL;
AT $E112: BYTE  DDH(0),  DATAH,  CONH;

PROCEDURE  PIA SETUP;
   CONL = $00; /* PREPARE TO ADDRESS DATA DIR */
   CONH = $00;
   DDL  = $FF; /* SET FOR OUTPUT */
   DDH  = $FF;
   CONL = $04; /* SELECT SIDE A AS OUTPUT NO INTERRUPTS
*/
   CONH = $2C; /* SELECT SIDE B AS OUT WITH CB22 AD
OUTPUT */
ENDPROC;

PROCEDURE UP;
  LBIT = $00;
  HBIT = $00;
  REPEAT
     REPEAT
        DATAL = LBIT;
        DATAH= HBIT;
        LBIT = LBIT + $01;
     UNTIL LBIT = $FF;
   DATAL = LBIT;
   DATAH = HBIT;
   HBIT = HBIT + $01;
   LBIT = $00;
   UNTIL HBIT =$10;
ENDPROC;

PROCEDURE RUN;
   PIA SETUP;
   REPEAT
     UP;
   FOREVER;
```

```
10 REM D/A CONV TEST PROGRAM  9/14/84 GIVES SAWTOOTH
OUT
20 REM FILE NAME DA-SAW3.BAS
22 REM USES THOMAS INSTRUMENTATION SP-1 BOARD AT $E100
24 REM BY J. A. MCDANIEL, UNIV. MAINE AT FARMINGTON
30 REM USES PIA 11 ADDRESSEO AT DATA A $E110; CONTROL A
$E111
40 REM DATA B $E112; CONTROL B $E113
45 POKE HEX ("E111"),0:POKE HEX("E113"),0: REM PREPARE TO
SET
50 REM DATA DIRECTION REGISTERS
60 POKE HEX("E110"),HEX("FF"):POKE HEX("E112"),HEX("FF")
65 REM SET BOTH SIDES AS OUTPUT
70 POKE HEX("E111"),4: REM SET SIDE A AS OUTPUT NO
INTERRUPTS
90 POKE HEX("E113"),HEX("2C"): REM ENABLE CB22 AS OUTPUT
TO
100 REM CONTROL O/A NOT SELECT- MSB OF B USED AS NOT
WRITE
130 FOR N$ = 0 TO 15
140 FOR I$ = 0 TO 255
150 POKE HEX("E110"),I$
160 POKE HEX("E112"),N$
180 NEXT I$
190 NEXT N$
200 GOTO 130
220 END
```

A/D CONVERTER

The design for the A/O converter was taken from the applications literature of National semiconductor with the analog switching, +/- 10 volt amplifiers and interfacing to the 6821 designed by Ron. The reference and 0 to 20 millivolt amplifiers are of my design. The cost of the amplifiers may be significantly reduced by using cheaper amplifiers than the AD647KH specified, but I prefer the Analog Devices ones, for they always perform as expected and often require no trimming. The reference voltage for the A/D, the switches and the channel selectors is derived from the excellent 10.00 reference of the A0567KO. Figure 2 gives the diagram for tha 0 to 20 millivolt amplifier; Figure 3 gives the diagram for the A/D converter and the +/- 10 volt amplifie s.

The basic A/O converter is designed for a 0 to 5 volt input, and thus has a resolution of 1.221 millivolts. The analog amplifier-scalars to scale a +/- 10 volt, and 0 to 20 or +/- 10 millivolts (easily changed to other values) to tha requisite 0 to 5 volt range are described. BASIC programs are describad which: (1) makes an input of +/- 10 amplifier yield +2047 and -2048 respectively (RWA), and (2) makes an an input of 0 or 20 millivolts yield 0000 or 4095 respectively.

The basic calibration procedure is to apply a voltage of 5.0000 to pins 18 and 19 of the ADC1210HCD and then adjust the 5 volt reference supply until the LSB flickers aqually between 0 and 1 with all other bits being off. Then a voltage of 0.00061 volts is applied to pins 18 and 19 and the zero adjust (RI on Figure 3) is adjusted until the LSB flickers between 0 and 1 with all other bits being on. (The converter is operating in the complementary binary output mode, and a short PL9 program is given below which outputs a hex word equal to the converter output, the inverse of the hex word, and the actual voltage input on a 0 to 5 basis.

After the A/D has been calibrated, the sample and hold should be connected and zeroed. With the AD647K no zeroing was really necessary as the amplifiers can be offset slightly to compensate it. (If it is necessary to zero the sample and hold, a zeroing circuit like the one on the +/- 10 volt amplifier may be added. But notice that the sample and hold then has a gain of 1.02 as does the +/- 10 volt amplifier.)

The +/- 10 volt amplifiers can be zeroed by inputting 0.0000 and setting the output to 0 using Ron's program. The actual output will be 2.500 volts at the amplifier.

The 0 to 20 millivolt amplifiers may be set up as follows: with 0.0000 volts in, adjust R8 until the output of the first amplifie is 0.0000 as indicated by the voltmeter. (If exactly 0 can not be attained, then set the output so that it is slightly negative.) Then adjust R9 until the output of the second amplifier is 0.0000. If the output of the first amplifier is not 0, then that is permissible, but the output of the final amplifier should be set to -10 times the output attained by the first by adjustment of R9. After these adjustments are finished, R10 should be connected to the -15 volt suppiy and adjusted until the output is 2.500 volts. The amplifier is calibrated for a 0 to 20 or +/- 10 millivolt input depending on whether R10 is connected or not. If the amplifier is only needed for + voltages, then omit R10 and the associated resistors.

```
10 REM DRIVER FOR 12 BIT A/D CONVERTER IN BASIC
20 REM
25 REM  **  BY R. W. ANDERSON  **
30 REM  **  ANN ARBOR, MICH    **
35 REM  **************************
40 REM PORT ADDRESS IS $E100 ON SAMPLE PROGRAM,
50 REM BUT MAY BE CHANGED TO ANY CONVENIENT
60 REM I/O ADDRESS-- CV IS Variable FOR CHANNEL
65 REM NUMBER OF A/D
70 GOTO 310:  REM SKIP SUBROUTINES
80     REM
90      REM FIRST THE PORT INITIALIZE SECTION
100 POKE HEX ("E110"),0 : POKE HEX ("E11F"),0
110 POKE HEX ("E11C"),0 : POKE HEX ("E11E"),HEX ("FO")
120 POKE HEX ("E11D"),4 : POKE HEX ("E11F"),HEX ("3C")
130 RETURN                : REM THIS IS A SUBROUTINE
140      REM
150      REM  THE CONVERT SUBROUTINE
160 CV$ = CV$ * 16                 : REM LEFT SHIFT FOUR
PLACES
170 POKE HEX ("E11E"),CV$        : REM SET UP CHANNEL
NUMBER
180 OA$ = PEEK(HEX("E11C"))    : REM CLEARS CONVERSION
185 REM                            COMPLETE FLAG
190 POKE HEX ("E11F"),HEX("34") : REM TURN ON CONVERT
PULSE
200 POKE HEX ("E11F"),HEX("3C") : REM TURN CONVERT PULSE
OFF
210 ST$ = PEEK(HEX("E110"))    : REM READ STATUS
220 IF ST$ < 128 THEN 210        : REM WAIT UNTIL HI BIT IS
ON
230 OA$=(PEEK(HEX("E11E")) AND 15) * 256 +
PEEK(HEX("E11C"))
240 IF OA$ > 2047 THEN DA$ = DA$ - 4096
245 REM ADJUST FOR NEGATIVE VALUES
250 RETURN
260  REM NOW YOU MAY ADO THE VALUE OBTAINED TO A
270  REM REAL VARIABLE FOR SUMMING AND AVERAGING
280  REM PROGRAMS.
290  REM
300  REM TEST PROGRAM
310 GOSUB 100
340 CV$ = 0 : REM SET CHANNEL TO ZERO
350 GOSUB 160
360 PRINT OA$
365 GOTO 340
370 END
```

```
10      REM DRIVER FOR 12 BIT A/D  CONVERTER IN BASIC
16      REM D-5 V INPUT 0=0000,  5=4095
20      REM PORT ADDRESS IS $E100 ON SAMPLE PROGRAM, BUT
30      REM MAY BE CHANGED TO ANY CONVENIENT 1/0 ADDRESS
40      REM CV$ IS Variable FOR CHANNEL NUMBER OF A/D
100     REM TEST PROGRAM ** BY J. A. MCDANIEL **
110 GOSUB 210: REM INITIALIZE PORT
120 INPUT " THE CHANNEL NUMBER IS ", CV$
125 CV$ = CV$ * 16: REM CALCULATE CHANNEL NUMBER
130 GOSUB 320: REM PERFORM CONVERSION
140 PRINT OA$
150 GOTO 130
160 END
170     REM
```

```
            BYTE MS_BITS;


AT $E11C:   BYTE PIA ADD(0), PIA AD, PIA AC;
AT $E11E:   BYTE PIA_BDD(0), PIA_BD, PIA_BC;
AT $E110:   BYTE PIADDL(0), PIADATAL, PIACONL;
AT $E112:   BYTE PIADOH(0), PIADATAH, PIACONH;

CONSTANT TRUE=-1,FALSE=0, MEM=$0000;
INCLUDE 0.IOSUBS;
INCLUDE 0.HEXIO.LIB;
INCLUDE REALCON;
```

```
PROCEDURE ESCAPE: BYTE FLAG;
  IF GETKEY = $1B /* ESCAPE KEY ON KEYBOARD */
    THEN FLAG = 1;
    ELSE FLAG = 0;
ENDPROC FLAG;

PROCEDURE SETUP AD; /* PROCEDURE TO SET UP A/D PIA */
   PIA AC = $00; /* PREPARE TO ADDRESS */
   PIA BC = $00; /* DATA DIRECTION REGISTERS */
   PIA ADD = $00; /* SET SIDE A AS INPUT */
   PIA BDD = $F0; /* SET 1/2 B AS INPUT 1/2 AS OUTPUT */
   PIA AC = $04;
   PIA BC = $3C;
ENDPROC;

PROCEDURE AD CONVERT(BYTE CHAN NU):
              BYTE COM B, TEMP, TEST B:
               INTEGER INT B, INT A, DUMMY:
   CHAN NU = SHIFT(CHAN NU,4);   /* LEFT SHIFT FOUR
PLACES */
   PIA BD = CHAN NU; /* SET UP CHANNEL NUMBER OF PIA */
   DATA A = PIA AD; /* CLEAR CONVERSION COMPLETE FLAG */
   PIA BC = $34; /* TURN ON AD CONVERT PULSE */
   PIA BC = $3C; /* TURN OFF AD CONVERT PULSE */

     REPEAT
        IF ESCAPE <> 0 THEN JUMP $0000;
        TEST B = PIA AC;
     UNTIL TEST B >= $80;

   DATA A = PIA AD
   DATA B = PIA BD AND $0F; /* REMOVE CHANNEL NU AND GET
4 MSBS */
   COM B = NOT(DATA B) AND $08;/*COMPLEMENT
UNCOMPLEMENTED BIT (MSB)*/
   TEMP = DATA B AND $07;/* GET COMPLEMENTED 3 BITS OF
MSB'S */
   MS BITS = COM B OR TEMP; /* PUT 4 MSB'S TOGETHER */
   INT B = SHIFT(INTEGER(MS BITS),8);
   INT A = INTEGER (DATA A);
   AD OUT = INT B OR INT A;
ENDPROC;

PROCEDURE PRNUM(REAL NUM):BYTE BUF(20);
   PRINT(ASCII(NUM,.BUF));
ENDPROC;

PROCEDURE STALL (REAL INDEX2): REAL COUNT;
   REPEAT
      COUNT = COUNT + 1;
      IF ESCAPE THEN JUMP $0000;
   UNTIL COUNT = INDEX2;
ENDPROC;

PROCEDURE TEST CONVERT(BYTE CHAN NU): BYTE INDEX:
   REAL A1, INDEX2, AVG, A AVG;
   SETUP AD;
   CRLF;
   PRINT("INPUT THE CHANNEL NUMBER "); CRLF;
   CHAN NU = GET HEX BYTE;
   CRLF;
   PU  HEX BYTE(CHAN NU);
   CRLF;
   REPEAT
      INDEX = 0;
      A AVG =0.0;
      REPEAT
         AD CONVERT(CHAN NU);
         PUT HEX ADDRESS(AD OUT);
         SPACE(5);
         PUT HEX ADDRESS(NOT(AD OUT) AND $0FFF);
         A1 = FLOAT(AD OUT);
         A1 = A1 / 4096-*5
         SPACE(5);
         PRNUM(A1);
         CRLF;
         A AVG = A AVG + A1;
         INDEX = INDEX + $01;
      UNTIL INDEX = $15;
      AVG = A AVG/21;
      SPACE(10); PRNUM(AVG);
      CRLF;
      INDEX2 = 3000;
      STALL(INDEX2);
      CRLF;
   FOREVER;
```

CAUTION:

Both the +/- 10 volt amplifier and particularly the 0 to 20 millivolt amplifier can under overload conditions place +/- 10 to 12 volts on the multiplexer input. Under these conditions no damage will result; however, the switches no longer properly work and channels which are supposedly off will feed into the one which is on, causing erroneous results.

CONSTRUCTION HINTS:

A few construction hints might be helpful at this point: all the resistors should be 1% metal film, and the feedback components in the 0-20 millivolt and the +/- 10 volt amplifiers should be matched to 0.1% or better. The 0.1 ufd integrating capacitor should be polystyrene or polyester definitely not ceramic. The bypass capacitors should be ceramic and miniature tantalum for the 0.1 and 10 ufd., respectively.

I found it very helpful to separate the analog grounds (as much as possible) from the SP-1 board. These grounds are symbolized by the triangles on the diagrams and were all tied to a piece of 1/8 inch shield braid run along the top of the board by the use of # 22 or #20 wire with most of them having their own wire. The shield braid was covered by plastic insulation and run to the interconnection board where the three main filter capacitors of my Gimix tie together. The braid was attached by a solder lug placed under a capacitor ground screw. The digital grounds were all run together on the SP1 board.

It is also helpful to keep the clock divider somewhat removed from the analog circuitry particularly the trimmer potentiometers in order to avoid injecting digital noise into the reference lines. It might also be helpful to shield the analog reference, zeroing, and analog input lines to the A/D as I have experienced digital feed into some of them. The coupled voltages are small but enough to affect 12 bit accuracy and are practically impossible to pinpoint by a scope. Under unfavorable conditions merely connecting the ground lead to the scope can make the error worse by a factor of 10 from injected hum. If you use a scope to detect millivolt fluctuations you will want to float the scope from the AC power line and run a ground to the end of the shield braid that connects to the filter capacitors in the computer. (The scope probe doesn't make a good enough ground for my scope and a wire should be directly connected.)

Finally, to help conserve space on the board, the channel input resistors can be placed so that the leads are only 0.1 inch apart if every other one is first inserted 0.2 inches apart and then the remainder inserted so that they rest on top of the first row. I would also recommend that you place the A/D converter on the right hand side of the SP-1 board since there is a larger square area on the right as opposed to the long thin more or less rectangular area on the left. This would mean that the A/D PIA should be IC 11 on the SP-1 board.

- - -

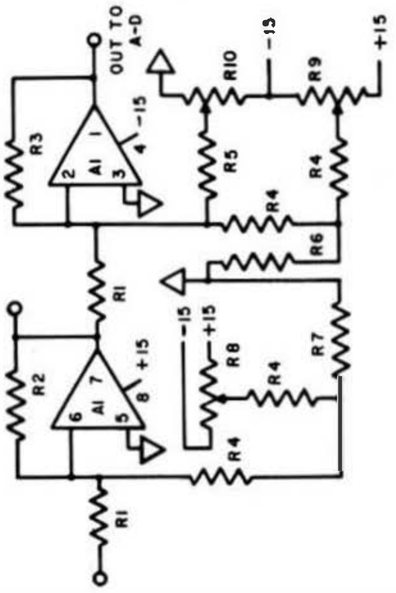


FIGURE 2. 0-20 MV AMPLIFIER
R1 10K, R2 200 K, R3 100K, R4 634K, R5 511 K, R6 2K,
R7 1.5 K, R8, R9, and R10 20 K 15 Turn Trimmers
A1 Analog Devices AD 647KH

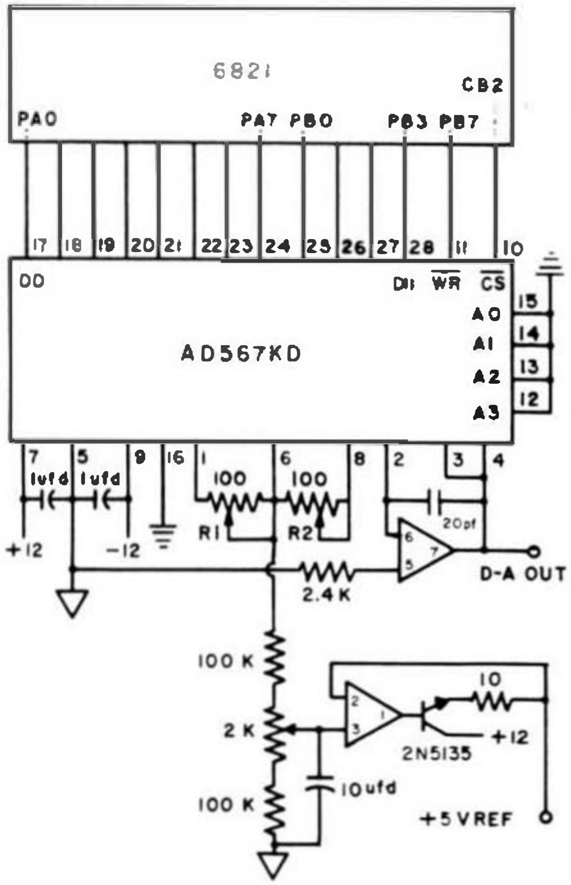


FIGURE 1. DIGITAL TO ANALOG CONVERTER
OP AMP Analog Devices AD647KH
R1, R2 100 ohm 15 Turn Trimmers (SEE TEXT)

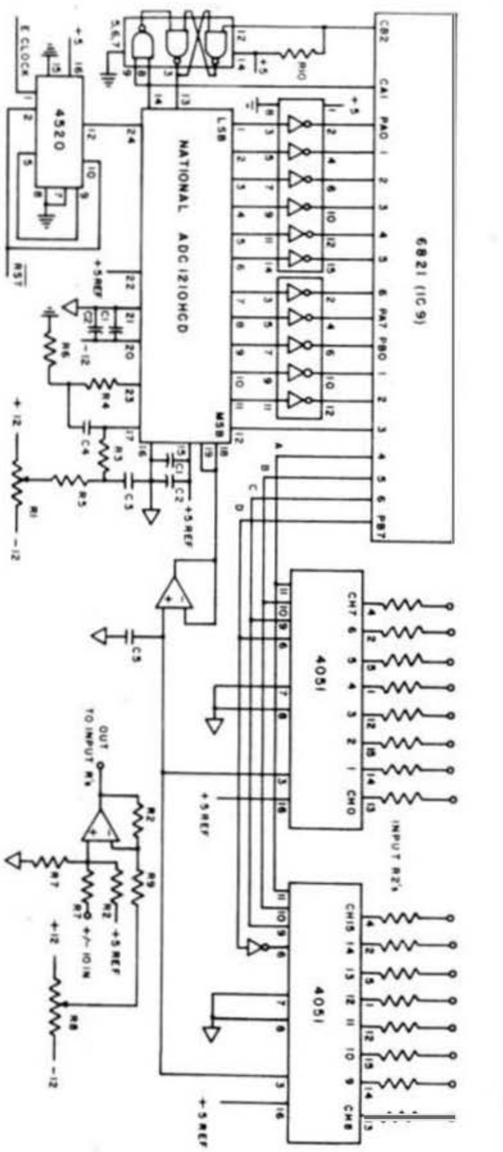


FIGURE 3. ANALOG TO DIGITAL CONVERTER AND +/- 10 VOLT AMPLIFIER
R1 20K TRIMMER, R2's 10K, R3 634K, R4 200K, R5 316K, R6 1K, R7 20K, R8 10K TRIMMER, R9 510K, R10 47K
C1 0.1ufd, C2 10ufd C3 10ufd, C4 220pf, C5 0.1ufd, INVERTERS 4049, AMPLIFIERS AD647KH, T1 TL084, or T1 TL082
NAND's 4011

BIT BUCKET

MICRONICS
RESEARCH CORP.
Microcomputers - Hardware and Software
GIMIX® Sales, Service and Support

21 December 1984

68 Micro Journal,
5900 Cassandra Smith,
Hixson, TN 37343

Dear Don,

Some time ago - I've forgotten in which issue of 68 Micro - one of your readers submitted an enhancement to Leo Taylor's COPY utility to output Carriage-Returns to his printer. He also suggested it would be nice if it could handle missing File-Numbers when the 'F' option was in use so that if, say, the range 2-15 were being copied and File #7 were missing from the sequence, the COPY procedure would not be aborted. Well, I accepted the challenge and am enclosing the necessary patch to implement his suggestion. Thanks for the idea!

1. Locate the label SKPFIL in the assembly-listing. Six lines above this, change BMI BADNO to BMI NOFILE.
2. Append the following code to SKPFIL, immediately after the instruction BRA RALOOP :

```
   NOFILE INC  LORANG+1 Adjust File #
          BNE  NOFIL1
          INC  LORANG
   NOFIL1 LDX  #NOFIL Missing File message
          JSR  PSTRNG
          CLRB
          LDX  #LORANG Display File #
          JSR  OUTDEC
          LDX  #SKPDIR+22 Continue?
          JSR  ASKMSG
          BEQ  RALOOP Yes
          CLR  PNTDRV No
          JMP  EXIT2
```

3. Insert the following in the messages section, immediately above WAITMS :

```
   NOFIL FCC "Non-existent File #"
         FCB 4
```

This letter has set me to thinking that if it were not for seeing our friend's suggestion, this idea might not have occurred to me. Perhaps there are other readers out there with thoughts of "Wouldn't it be nice if a certain utility had such and such an option, or even if there were a utility to do whatever." but perhaps they don't have the expertise to carry out their ideas. I hereby invite you folks out there to contact me with your ideas, and maybe between us we can come up with something we can all use to our advantage.

Over the next few months I hope to submit upgraded versions of other utilities I've been working on. Have a merry Xmas and a VERY happy New Year.

33303 LYNN AVENUE,
ABBOTSFORD,
BRITISH COLUMBIA,
CANADA, V2S 1E2

Sincerely,

Bob Jones

R Jones
President

MOTOROLA INC.
MOS Integrated Circuits Group
75[illegible] BLUESTEM BLVD
AUSTIN, TEXAS 78721
For further information contact

PRESS INFORMATION

Editorial Contact: Val Bauer

512 928-6804

Reader Contact: Beverly Sill

201 592-2348

INTRODUCTION TO INTEGRATED CIRCUIT LAYOUT

By Brian Spinks

Austin, Texas, December 1, 1984 — A definitive college-level textbook on integrated circuit layout is finally available. Introduction to Integrated Circuit Layout, by Brian Spinks, offers the basic theory and method of integrated circuit design to engineering and drafting students as well as other technologists, as a preparation for integrated circuit (IC) mask design.

Introduction to Integrated Circuit Layout is intended to provide the student with a working vocabulary of the trade, the basic theory necessary for the layout of metal-oxide-semiconductor integrated circuits (MOS ICs), and a method for translating a logic diagram to a schematic design for use in designing an integrated circuit. The reader is also shown techniques for the design of a composite drawing of masks for use in the fabrication of ICs, and the requirements for noncircuit elements of ICs, such as logos, alignment keys, and etch marks.

-more-

The material for Introduction to Integrated Circuit Layout was developed from a second-year college-level drafting course. In 1979, Brian Spinks began compiling notes for a technical course designed to alleviate the necessity for in-house training in integrated circuit design. The lack of available teaching materials led to the eventual publication of Introduction to Integrated Circuit Layout. The book is addressed to students possessing some familiarity with printed-circuit layout and design, electronic schematic drafting, and basic electronics.

Introduction to Integrated Circuit Layout contains a very useful glossary of the trade vocabulary, as well as easy-to-read drawings, diagrams, examples, and student exercises where appropriate. The material is compiled solely from sources provided by present and former members of the Microprocessor Products Division Design Staff, Motorola, Inc., Austin, Texas.

Brian Spinks received his BA in Mathematics in 1962 from the University of Texas at Austin, where he also worked at the Defense Research Laboratory. In 1977, he was awarded a BSEE from the University of Houston. Mr. Spinks has worked at Lockheed Electronics on a NASA Program, at Texas Instruments, and at American Micro Systems, Inc., where he worked for two years before joining the Applications Engineering Staff at Motorola in 1975.

Introduction to Integrated Circuit Layout, by Brian Spinks, 1985, is $19.95 in soft cover, $24.95 in hard cover. For further information regarding Introduction to Integrated Circuit Layout, please contact Beverly Sill, College Publicity, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 07632, or call collect, (201) 592-2348.

EDITORIAL CONTACT: Ed Prestwood
(602) 994-6959

MOTOROLA PUBLISHES NEW 16/32-BIT MICROCOMPUTER BOARD DATA BOOK

FOR IMMEDIATE RELEASE ... A new data book covering the broad range of Motorola 16/32 Bit Microcomputer Board-Level products is now available.

Chapters cover VMEmodules, VERSAmodules, I/Omodules, operating systems, development systems, system bus technical summaries, and customer support. Technical specifications, photos, charts, and graphs are used to describe a variety of board, system, and software products. The board-level products are based on the M68000 Microprocessor Family, and include a monoboard microcomputer utilizing the MC68020—Motorola's new, full 32-bit MPU.

A copy may be ordered by requesting DL127 from Motorola Literature Distribution Center, 616 W. 24th Street, Tempe, Arizona 85282, (602) 994-6561. Price is $2.05.

###

Note to Editors: Please do not publish without including price information.

MOTOROLA
Semiconductor Products Inc.
P.O. BOX 20912 PHOENIX, ARIZONA 85036

Innovative systems
through silicon.

Plexus
Computers,
Inc.

3833 North First Street
San Jose, CA 95134
408 • 943 9433
TWX/TELEX 910 338 2223

News Release

Contact: Jeff Stives
Marketing Communications
408 • 943 2247

FOR RELEASE ON OR AFTER
14 NOVEMBER 1984

PLEXUS ENTERS 1-8 USER MARKET WITH MC68010-BASED $10,000 UNIX SUPERMICRO COMPUTER

SAN JOSE, CA., NOV. 14, 1984 — Plexus Computers, Inc. today announced its new entry into the 1-8 user marketplace with a high performance MC68010-based supermicro computer specifically designed to run the UNIX* operating system.

Designated the Plexus P/15, the new 32-bit supermicro offers full compatibility with the rest of the Plexus UNIX-based line of products and is targeted at the growing VAR/OEM market, as well as toward large end-users.

As with the other Plexus computers, the P/15 features a multiprocessor architecture and utilizes two MC68010 microprocessors, a memory capacity of up to 2 Mbytes and eight full duplex serial ports for terminals or other peripherals.

"The P/15 is aimed directly at the small user who needs not only 8 user capacity, but high performance as well," said Edward J. Hayes, Plexus' VP of Marketing.

"There is a very large segment of the OEM/VAR market that has indicated a need for the P/15," said Hayes. "In most cases, these VAR's have been selling a system allowing from one to four users and they have been running out of steam...out of power as their application needs grew. The P/15, with its superior multiprocessor architecture utilizing two of the more powerful MC68010 microprocessors, is offered as a solution to this VAR dilemma."

Hayes pointed out that the P/15 is an entirely new product, designed by Plexus engineers especially for this market.

"As the Plexus product line continues to expand, new products like the P/15 will appear; not as a replacement for existing systems, but as viable new alternatives for VAR's and end-users who want UNIX* and want powerful, multi-tasking supermicros," Hayes said.

The P/15 is packaged to fit in a very limited space often found in the typical business office. With its attractive low profile design, a height of under 25 inches and a weight of under 75 pounds, it easily fits along side other office equipment and furniture. The P/15 is totally self-contained, allowing for up to 54 Mbytes of disk storage in 2 Winchester-type disks, plus a single 5.25 inch double sided, double density floppy disk and uses standard 115VAC power.

P/15 users should expect significantly faster response time in the multiuser environment than they are accustomed to seeing in other similarly priced machines, Hayes pointed out. He indicated that these benefits are a factor of both the multiprocessor design and the separate 32-bit processors for I/O functions, such as data communications, disk subsystem control and functions such as job processing and operating system execution.

The I/O processor is a 32-bit device controlling an SCSI subsystem interface offering full error-correction facilities as well as full SCSI interface functions serving the hard and floppy disks. In addition, the I/O processor handles all character I/O to and from the eight serial ports removing this overhead from the job processor.

The Plexus P/15 CPU job processor, using the 32-bit MC68010 with a 10MHz clock, operates with no wait states, through a shared high-speed map, with 8 Mbytes of address space. It supports the IEEE proposed standards for floating point arithmetic.

Utilizing current 256K-bit RAM devices, the P/15 memory is available with up to 1 million 16-bit words.

The P/15 is available within 120 days A.R.O. and is priced at $10,950 US, quantity one, with .5 Mbyte of RAM and 12 Mbytes of Winchester disk storage, plus a 1 Mbyte floppy disk drive.

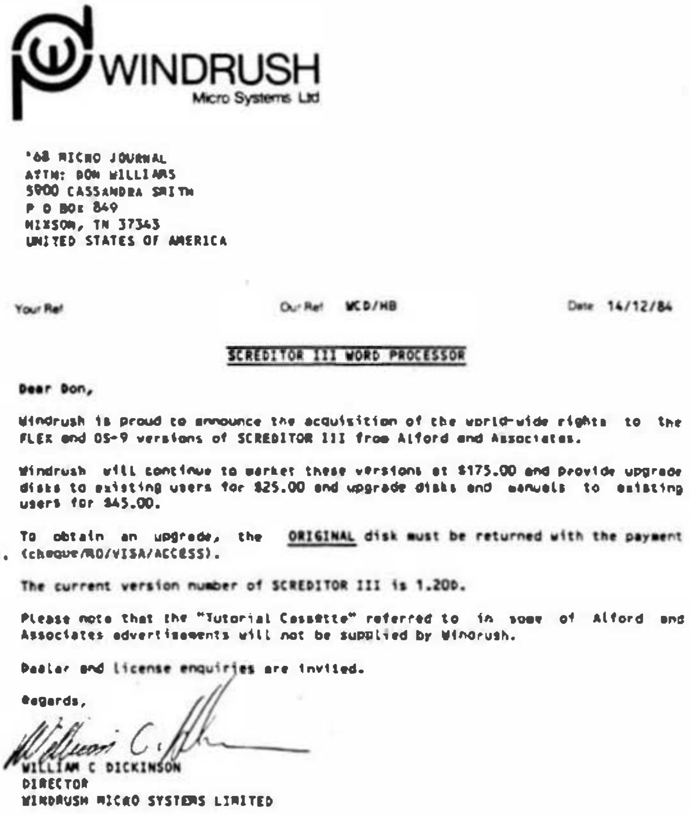
Plexus Computers, Inc. manufactures a fully compatible line of high performance 32-bit supermicrocomputers designed for the VAR/OEM and large end-user, utilizing the UNIX* operating system and a unique multiprocessor architecture which provides users with exceptional power and performance. Plexus markets its products through sales offices in the U.S. and via its distributors and subsidiaries throughout the world.

For additional information, please contact Jeff Stives, Plexus' Marketing Communications Manager, 3833 North First Street, San Jose, CA. 95134, (408)943-2247.

- 30 -

1984

*UNIX is a trademark of Bell Laboratories.



WINDRUSH
Micro Systems Ltd

'68 MICRO JOURNAL
ATTN: DON WILLIAMS
5900 CASSANDRA SMITH
P O BOX 849
HIXSON, TN 37343
UNITED STATES OF AMERICA

Your Ref Our Ref WCD/HB Date 14/12/84

SCREDITOR III WORD PROCESSOR

Dear Don,

Windrush is proud to announce the acquisition of the world-wide rights to the FLEX and OS-9 versions of SCREDITOR III from Alford and Associates.

Windrush will continue to market these versions at $175.00 and provide upgrade disks to existing users for $25.00 and upgrade disks and manuals to existing users for $45.00.

To obtain an upgrade, the **ORIGINAL** disk must be returned with the payment (cheque/MO/VISA/ACCESS).

The current version number of SCREDITOR III is 1.20D.

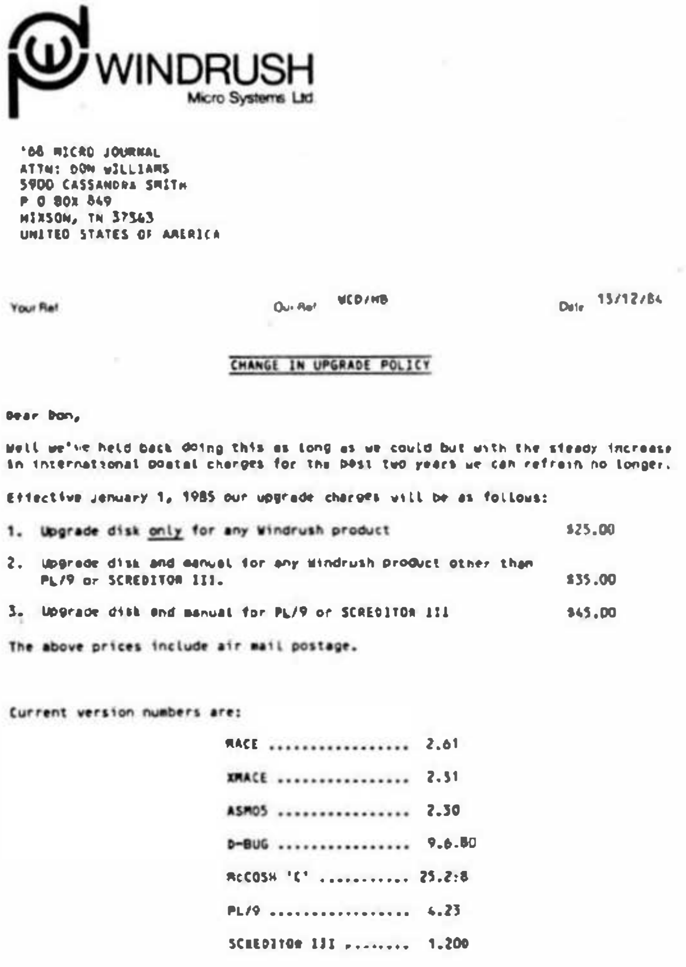
Please note that the "Tutorial Cassette" referred to in some of Alford and Associates advertisements will not be supplied by Windrush.

Dealer and license enquiries are invited.

Regards,

William C. [illegible]

WILLIAM C DICKINSON
DIRECTOR
WINDRUSH MICRO SYSTEMS LIMITED



WINDRUSH
Micro Systems Ltd

'68 MICRO JOURNAL
ATTN: DON WILLIAMS
5900 CASSANDRA SMITH
P O BOX 849
HIXSON, TN 37343
UNITED STATES OF AMERICA

Your Ref Our Ref WCD/HB Date 13/12/84

CHANGE IN UPGRADE POLICY

Dear Don,

Well we've held back doing this as long as we could but with the steady increase in international postal charges for the past two years we can refrain no longer.

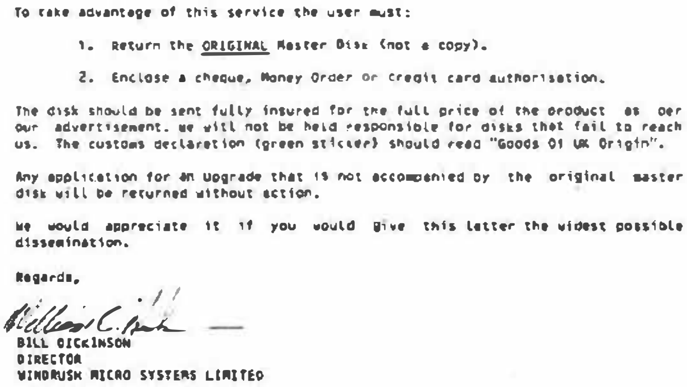
Effective January 1, 1985 our upgrade charges will be as follows:

1.	Upgrade disk only for any Windrush product	$25.00
2.	Upgrade disk and manual for any Windrush product other than PL/9 or SCREDITOR III.	$35.00
3.	Upgrade disk and manual for PL/9 or SCREDITOR III	$45.00

The above prices include air mail postage.

Current version numbers are:

MACE	2.61
XMACE	2.51
ASM05	2.30
D-BUG	9.6.B0
McCOSH 'C'	25.2:8
PL/9	4.23
SCREDITOR III	1.20D



To take advantage of this service the user must:

1. Return the ORIGINAL Master Disk (not a copy).

2. Enclose a cheque, Money Order or credit card authorisation.

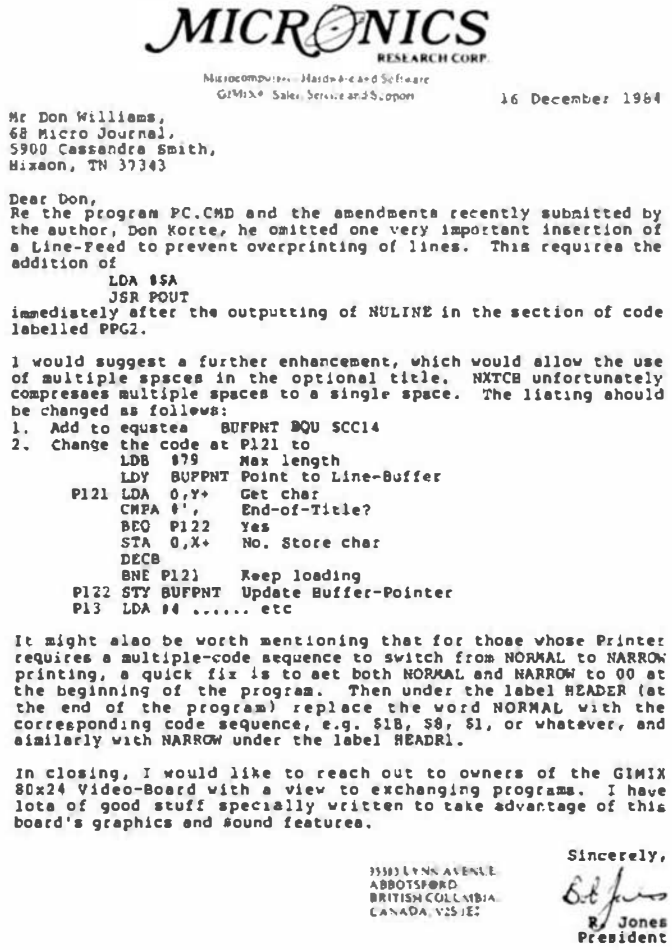
The disk should be sent fully insured for the full price of the product as per our advertisement. We will not be held responsible for disks that fail to reach us. The customs declaration (green sticker) should read "Goods Of UK Origin".

Any application for an Upgrade that is not accompanied by the original master disk will be returned without action.

We would appreciate it if you would give this letter the widest possible dissemination.

Regards,

BILL DICKINSON
DIRECTOR
WINDRUSH MICRO SYSTEMS LIMITED



MICRONICS
RESEARCH CORP.

Microcomputers Hardware and Software
GIMIX® Sales Service and Support

16 December 1984

Mr Don Williams,
68 Micro Journal,
5900 Cassandra Smith,
Hixson, TN 37343

Dear Don,
Re the program PC.CMD and the amendments recently submitted by the author, Don Korte, he omitted one very important insertion of a Line-Feed to prevent overprinting of lines. This requires the addition of

```
        LDA $SA
        JSR POUT
```

immediately after the outputting of NULINE in the section of code labelled PPG2.

I would suggest a further enhancement, which would allow the use of multiple spaces in the optional title. NXTCH unfortunately compresses multiple spaces to a single space. The listing should be changed as follows:

```
1.  Add to equates   BUFPNT EQU $CC14
2.  Change the code at P121 to
          LDB  $79    Max length
          LDY  BUFPNT Point to Line-Buffer
     P121 LDA  0,Y+   Get char
          CMPA $',    End-of-Title?
          BEQ  P122   Yes
          STA  0,X+   No. Store char
          DECB
          BNE P121    Keep loading
     P122 STY BUFPNT  Update Buffer-Pointer
     P13  LDA $4 ...... etc
```

It might also be worth mentioning that for those whose Printer requires a multiple-code sequence to switch from NORMAL to NARROW printing, a quick fix is to set both NORMAL and NARROW to 00 at the beginning of the program. Then under the label HEADER (at the end of the program) replace the word NORMAL with the corresponding code sequence, e.g. $1B, $8, $1, or whatever, and similarly with NARROW under the label HEADR1.

In closing, I would like to reach out to owners of the GIMIX 80x24 Video-Board with a view to exchanging programs. I have lots of good stuff specially written to take advantage of this board's graphics and sound features.

Sincerely,

R. Jones
President

33503 LYNN AVENUE
ABBOTSFORD
BRITISH COLUMBIA
CANADA, V2S 1E2

The BASIC programs DATACQ, DATA8, and DATA16 work with an AD-16 JPC analog to digital conversion board. USER subroutines are called by the BASIC programs to insure that as soon as the conversion has been completed the next step in the data taking process can be initiated. The machine language routines disable the IRQ and allow a complete handshake between the ADC0817 A/D chip and the 6821 PIA by checking for the status of the eighth bit of the 6821's status register. The manner of handshaking provided in the machine language program can be used to form the nucleus of a more complex program.

Maximum sampling rate for one channel is 9000 samples per second for a 1 MegaHertz clock 6800 based system where the 7474 chip has been removed from the AD-16 board and pins 3 and 5 of the 7474 socket are tied together in order to clock the ADC0817 at 1 MegaHertz. A 1000 Hertz 5.0 volt peak to peak sine wave with a DC offset of +2.5 volts was found to be reproducible at 9000 samples per second.

It is worth noting that in all the USER machine language routines, where more than one channel is selected, there is a delay of about 24 clock cycles. This serves to insure that the correct channel is accessed by keeping the ALE and Start of Conversion pins of the ADC0817 high for the duration of the delay. BASIC has been used to display the data and engineering units and other formats in BASIC are possible. The programs provided are essentially test programs to tell the user that his A/D board is functioning or not. DATA16 samples 16 channels in numerical order one channel after another for a total of 16 samples per channel. DATA8 allows the user to select eight channels in any sampling order. Each channel is sampled one channel after another for a total of thirty-two samples per channel. DATACQ allows the user to select any one of sixteen channels with 256 samples being taken for the channel chosen. The gain for all channels is one. The AD-16 is on port 4.

```
10 REM THE NAME OF THIS PROGRAM IS DATACQ
20 POKE HEX("26"),0
30 POKE HEX("27"),0
40 LET S=0
50 EXEC,"GET,USER3.BIN"
60 PRINT "DO YOU WANT HARDCOPY - 'Y' FOR YES OR 'N' FOR NO."
70 INPUT P$
80 IF P$<>"Y" AND P$<>"N" THEN GOTO 60
90 PRINT "ENTER THE CHANNEL YOU WANT SAMPLED"
100 INPUT C
110 IF C<1 THEN GOTO 90
120 IF C>16 THEN GOTO 90
130 IF C=1 THEN Y=16
140 IF C=2 THEN Y=17
150 IF C=3 THEN Y=18
160 IF C=4 THEN Y=19
170 IF C=5 THEN Y=20
180 IF C=6 THEN Y=21
190 IF C=7 THEN Y=22
200 IF C=8 THEN Y=23
210 IF C=9 THEN Y=24
220 IF C=10 THEN Y=25
230 IF C=11 THEN Y=26
240 IF C=12 THEN Y=27
250 IF C=13 THEN Y=28
260 IF C=14 THEN Y=29
270 IF C=15 THEN Y=30
280 IF C=16 THEN Y=31
290 POKE HEX("5999"),Y
300 POKE HEX("24"),HEX("60")
310 POKE HEX("25"),HEX("00")
320 LET A=USR(0)
330 IF P$="Y" THEN OPEN "0.PRINT" AS 0
340 IF P$="Y" THEN PRINT #0,"
350 FOR I=1 TO 256
360 LET Y=PEEK(HEX("7000")+I)
370 LET Y$=STR$(Y)
380 LET T=LEN(Y$)
390 IF P$="Y" THEN GOTO 420
400 PRINT SPC(5-T);Y;
410 GOTO 460
420 PRINT #0,SPC(5-T);Y;
430 LET S=S+1
440 IF S=8 THEN PRINT #0
450 IF S=8 THEN S=0
460 NEXT I
470 IF P$="Y" THEN CLOSE 0
480 END
```

2 Jeffrey M. Craig Apt. 912 - 3001 S. King Dr. Chicago, IL 60616 4/6/82

```
1Ø REM THE NAME OF THIS PROGRAM IS DATA8
2Ø POKE HEX("26"),Ø
3Ø POKE HEX("27"),Ø
4Ø LET N=Ø
5Ø LET S=Ø
6Ø EXEC,"GET,USER2.BIN"
7Ø PRINT "DO YOU WANT HARDCOPY - 'Y' FOR YES III 'N' FOR NO."
8Ø INPUT P$
9Ø IF P$<>"Y" AND P$<>"N" THEN GOTO 7Ø
1ØØ PRINT "ENTER THE EIGHT CHANNELS YOU WANT SAMPLED - "
11Ø FOR I=1 TO 8
12Ø INPUT Z
13Ø IF Z<1 THEN GOTO 1ØØ
14Ø IF Z>16 THEN GOTO 1ØØ
15Ø IF Z=1 THEN Y=16
16Ø IF Z=2 THEN Y=17
17Ø IF Z=3 THEN Y=18
18Ø IF Z=4 THEN Y=19
19Ø IF Z=5 THEN Y=2Ø
2ØØ IF Z=6 THEN Y=21
21Ø IF Z=7 THEN Y=22
22Ø IF Z=8 THEN Y=23
23Ø IF Z=9 THEN Y=24
24Ø IF Z=1Ø THEN Y=25
25Ø IF Z=11 THEN Y=26
26Ø IF Z=12 THEN Y=27
27Ø IF Z=13 THEN Y=28
28Ø IF Z=14 THEN Y=29
29Ø IF Z=15 THEN Y=3Ø
3ØØ IF Z=16 THEN Y=31
31Ø POKE HEX("5FF7")+I,Y
32Ø NEXT I
33Ø POKE HEX("24"),HEX("6Ø")
34Ø POKE HEX("25"),HEX("ØØ")
35Ø LET A=USR(Ø)
36Ø IF P$="Y" THEN OPEN "Ø.PRINT" AS Ø
37Ø IF P$="Y" THEN PRINT #Ø,"
38Ø FOR I=1 TO 256 STEP 8
39Ø GOSUB 72Ø
4ØØ NEXT I
41Ø PRINT
42Ø FOR I=2 TO 256 STEP 8
43Ø GOSUB 72Ø
44Ø NEXT I
45Ø PRINT
46Ø FOR I=3 TO 256 STEP 8
47Ø GOSUB 72Ø
48Ø NEXT I
49Ø PRINT
5ØØ FOR I=4 TO 256 STEP 8
51Ø GOSUB 72Ø
52Ø NEXT I
53Ø PRINT
54Ø FOR I=5 TO 256 STEP 8
55Ø GOSUB 72Ø
56Ø NEXT I
57Ø PRINT
58Ø FOR I=6 TO 256 STEP 8
59Ø GOSUB 72Ø
6ØØ NEXT I
61Ø PRINT
62Ø FOR I=7 TO 256 STEP 8
63Ø GOSUB 72Ø
64Ø NEXT I
65Ø PRINT
66Ø FOR I=8 TO 256 STEP 8
67Ø GOSUB 72Ø
68Ø NEXT I
69Ø PRINT
7ØØ IF P$="Y" THEN CLOSE Ø
71Ø END
72Ø LET Y=PEEK(HEX("6FFF")+I)
73Ø LET Y$=STR$(Y)
74Ø LET T=LEN(Y$)
75Ø IF P$="Y" THEN GOTO 78Ø
76Ø PRINT SPC(5-T);Y;
77Ø GOTO 85Ø
78Ø PRINT #Ø,SPC(5-T);Y;
79Ø S=S+1
8ØØ LET N=N+1
81Ø IF S=8 THEN PRINT #Ø
82Ø IF N=32 THEN PRINT #Ø
83Ø IF S=8 THEN S=Ø
84Ø IF N=32 THEN N=Ø
85Ø RETURN
```

```
1Ø REM THE NAME OF THIS PROGRAM IS DATA16
2Ø POKE HEX("26"),Ø
3Ø POKE HEX("27"),Ø
4Ø LET S=Ø
5Ø LET N=Ø
6Ø EXEC,"GET,USER3.BIN"
7Ø POKE HEX("24"),HEX("6Ø")
8Ø POKE HEX("25"),HEX("ØØ")
9Ø LET A=USR(Ø)
1ØØ PRINT "DO YOU WANT HARDCOPY? - 'Y' FOR YES III 'N' FOR NO."
11Ø INPUT P$
12Ø IF P$<>"Y" AND P$<>"N" THEN GOTO 1ØØ
13Ø IF P$="Y" THEN OPEN "Ø.PRINT" AS Ø
14Ø IF P$="Y" THEN PRINT #Ø,"
15Ø FOR I=1 TO 256 STEP 16
16Ø GOSUB 81Ø
17Ø NEXT I
18Ø PRINT
19Ø FOR I=2 TO 256 STEP 16
2ØØ GOSUB 81Ø
21Ø NEXT I
22Ø PRINT
23Ø FOR I=3 TO 256 STEP 16
24Ø GOSUB 81Ø
25Ø NEXT I
26Ø PRINT
27Ø FOR I=4 TO 256 STEP 16
28Ø GOSUB 81Ø
29Ø NEXT I
3ØØ PRINT
31Ø FOR I=5 TO 256 STEP 16
32Ø GOSUB 81Ø
33Ø NEXT I
34Ø PRINT
35Ø FOR I=6 TO 256 STEP 16
36Ø GOSUB 81Ø
37Ø NEXT I
38Ø PRINT
39Ø FOR I=7 TO 256 STEP 16
4ØØ GOSUB 81Ø
41Ø NEXT I
42Ø PRINT
43Ø FOR I=8 TO 256 STEP 16
44Ø GOSUB 81Ø
45Ø NEXT I
46Ø PRINT
47Ø FOR I=9 TO 256 STEP 16
48Ø GOSUB 81Ø
49Ø NEXT I
5ØØ PRINT
51Ø FOR I=1Ø TO 256 STEP 16
52Ø GOSUB 81Ø
53Ø NEXT I
54Ø PRINT
55Ø FOR I=11 TO 256 STEP 16
56Ø GOSUB 81Ø
57Ø NEXT I
58Ø PRINT
59Ø FOR I=12 TO 256 STEP 16
6ØØ GOSUB 81Ø
61Ø NEXT I
62Ø PRINT
63Ø FOR I=13 TO 256 STEP 16
64Ø GOSUB 81Ø
65Ø NEXT I
66Ø PRINT
67Ø FOR I=14 TO 256 STEP 16
68Ø GOSUB 81Ø
69Ø NEXT I
7ØØ PRINT
71Ø FOR I=15 TO 256 STEP 16
72Ø GOSUB 81Ø
73Ø NEXT I
74Ø PRINT
```

```
750 FOR X=16 TO 256 STEP 16
760 GOSUB 810
770 NEXT X
780 PRINT
790 IF P$="Y" THEN CLOSE 0
800 END
810 LET Y=PEEK(HEX("6FFF")+X)
820 LET Y$=STR$(Y)
830 LET T=LEN(Y$)
840 IF P$="Y" THEN GOTO 870
850 PRINT SPC(5-T);Y;
860 GOTO 940
870 PRINT #0,SPC(5-T);Y;
880 LET S=S+1
890 LET N=N+1
900 IF S=8 THEN PRINT #0
910 IF N=16 THEN PRINT #0
920 IF S=8 THEN S=0
930 IF N=16 THEN N=0
940 RETURN
```

```
 NAM USER1
 OPT PAG
 ORG $5000

DATREG EQU $8010
STATRG EQU $9011
ADDREG EQU $8012
CONTRG EQU $8013
GAINCH EQU $5999

 LDA A #$0
 STA A STATRG
 LDA A #$0
 STA A DATREG
 LDA A #$6
 STA A STATRG
 LDA A #$0
 STA A CONTRG
 LDA A #$FF
 STA A ADDREG
 LDA A #$34
 STA A CONTRG

 LDA A GAINCH
 STA A ADDREG
 LDX #$7000
BEGIN LDA A #$3C
 STA A CONTRG
 LDA A #$34
 STA A CONTRG
LOOP2 LDA A STATRG
AND A #%10000000
CMP A #%10000000
BNE LOOP2
LDA A DATREG
STA A 0,X
INX
CPX #$7101
BEQ EXIT
JMP BEGIN
EXIT RTS
 END
```

```
 NAM USER2
 OPT PAG
 ORG $6000

DATREG EQU $8010
STATRG EQU $8011
ADDREG EQU $8012
CONTRG EQU $8013
GAINC8 EQU $5FFF
GAINC7 EQU $5FFE
GAINC6 EQU $5FFD
GAINC5 EQU $5FFC
GAINC4 EQU $5FFB
GAINC3 EQU $5FFA
GAINC2 EQU $5FF9
GAINC1 EQU $5FF8

 LDA A #$0
 STA A STATRG
 LDA A #$0
 STA A DATREG
 LDA A #$6
 STA A STATRG
 LDA A #$0
 STA A CONTRG
 LDA A #$FF
 STA A ADDREG
 LDA A #$34
 STA A CONTRG
 LDX #$7000
CHAN1 LDA A GAINC1
 STA A ADDREG
 JSR HANDSK
CHAN2 LDA A GAINC2
 STA A ADDREG
 JSR HANDSK
CHAN3 LDA A GAINC3
 STA A ADDREG
 JSR HANDSK
CHAN4 LDA A GAINC4
 STA A ADDREG
 JSR HANDSK
CHAN5 LDA A GAINC5
 STA A ADDREG
 JSR HANDSK
CHAN6 LDA A GAINC6
 STA A ADDREG
 JSR HANDSK
CHAN7 LDA A GAINC7
 STA A ADDREG
 JSR HANDSK
CHAN8 LDA A GAINC8
 STA A ADDREG
 JSR HANDSK
 CPX #$7100
 BNE RETURN
 RTS
RETURN JMP CHAN1
HANDSK LDA A #$3C
 STA A CONTRG
 LDA A #$3
LOOP2 DEC A
 CMP A #$0
 BNE LOOP2
 LDA A #$34
 STA A CONTRG
LOOP3 LDA A STATRG
 AND A #%10000000
 CMP A #%10000000
 BNE LOOP3
 LDA A DATREG
 STA A 0,X
 INX
 RTS

 END
```

```
 NAM USER3
 OPT PAG
 ORG $6000

DATREG EQU $8010
STATRG EQU $8011
ADDREG EQU $8012
CONTRG EQU $8013
LDA A #$0
STA A STATRG
LDA A #$0
STA A DATREG
LDA A #$6
STA A STATRG
LDA A #$0
STA A CONTRG
```

```
 LDA A #$FF
 STA A ADDREG
 LDA A #$34
 STA A CONTRG

 LDX #$7000
LOOP1 CPX #$7100
 BEQ EXIT
 LDA B #16
LOOP2 STA B ADDREG
 JSR HANDSK
 INC B
 CMP B #32
 BEQ LOOP1
 JMP LOOP2
EXIT RTS
HANDSK LDA A #$3C
 STA A CONTRG
 LDA A #$3
LOOP3 DEC A
 CMP A #$0
 BNE LOOP3
 LDA A #$34
 STA A CONTRG
LOOP4 LDA A STATRG
 AND A #%11000000
 CMP A #%11000000
 BNE LOOP4
 LDA A DATREG
 STA A 0,X
 INX
 RTS

 END
```

FLEX Setime Utility

The following SETIME routine is almost an exact copy of SSB's DOS setime routines, except that it has been modified to operate under FLEX (since FLEX does not provide a routine to set the time for those of you who have the MM58167 on-board clock).

In addition to updating the real time clock, SETIME also modifies the FLEX date register appropriately, so that the DATE command need not be used se arately to set the month, day and year. One nice feature added to SETIME is the ability to set the year, and because it is a parameter on the command, it can be put in your startup file. By utilizing the FLEX date register to store the year (as the DATE command does), there is no need to have it hard-coded in the SETIME command. The format of the command is:

SETIME

The program will prompt you to enter the current date and/or time

SETIME D

This will give you the current time and date

SETIME 83

Change the year in FLEX's date register to "83"

SETIME 83 D

Set the year to "83" and display the date/time while you're at it

The real time clock for my 6809 system is at $F700 (label CLOCK as defined in the FLEX equates). Change this value appropriate! .

For a small fee of $12.00 (U.S), I will send you the source to the SETIME and EXTEND commands, as well as the FLEX equates, on a 5" floppy disk. Pleas specify whether you want it on 40 tracks or 80 tracks. Price includes cost of disk. Make check or money order payable to:

Scott Fraser
547 Sharron Bay
Winnipeg, Manitoba, Canada
R2G 0H8

```
                    *
                    * Setime - The SET TIME/DATE Transient allows the
                    *          user of SSB's Real Time Clock (RTC) to set
                    *          the time, date and year of the hardware
                    *          device.
                    *
                    * It is called as:
                    *
                    *           SETIME       (set time and date)
                    *
                    *           SETIME D     (display time & date)
                    *
                    *           SETIME YY    (set year given by "YY"
                    *                         digits)
                    *
                    *           SETIME YY D  (set year and display
                    *                         time & date)
                    *
                    *   NOTE: "YY" should be between 00 and 99
                    *
                    * This program does not use interrupts to update
                    * the time and date string on the display
                    * once a second; instead uses a software loop
                    * to determine a 1 cycle/sec interval
                    *
                    * This is a slight modification of DOS' SETIME cmd
                    * to work under FLEX (Sept, 1982)
                    *
               C387 ZRDTIM  EQU   RDTIME    "read time" rtn
               F80C PDATA   EQU   $F80C     addr of PDATA rtn (SSB MON)
                    *
                    * MM58167 register offsets
                    *
               0002 SEC     EQU   2         seconds
               0003 MIN     EQU   3         minutes
               0004 HOUR    EQU   4         hours

               0005 DOW     EQU   5         day of week
               0006 DOM     EQU   6         day of month
               0007 MON     EQU   7         month of year

C100                        ORG   UCA
C100 20    23               BRA   START
               0002 VN      EQU   2         version 2
C102                YDATE   RMB   16        holds the date
C112                YTIME   RMB   16        holds the time
C122                XTEMP   RMB   3

               C125 START   EQU   *
C125 108E F700              LDY   #CLOCK
C129 BD    CD27             JSR   NXTCH     get a character
C12C 25    34               BCS   START1    branch if nothing
C12E 81    44               CMPA  #'D       just want date/time?
C130 27    22               BEQ   DISTD     branch if so
                    *
                    * assume a year has been specified
                    *
C132 BD    CD27             JSR   NXTCH     get 1's digit in year
C135 25    2B               BCS   START1    branch if nothing
C137 1F    89               TFR   A,B       A -> B
C139 B6    CC19             LDA   PREVC     get 10's digit
C13C 17    02C9             LBSR  ASCBIN    convert to binary
C13F 25    4C               BCS   ERR1      leave if error
C141 B7    CC10             STA   SYDR+2    save year
C144 BD    CD27             JSR   NXTCH     anything else?
C147 81    20               CMPA  #$20      this a space?
C149 26    48               BNE   XIT       nope, then leave
C14B BD    CD27             JSR   NXTCH     go past spaces
C14E 25    43               BCS   XIT       if nothing, leave
C150 81    44               CMPA  #'D       want date/time?
C152 26    3F               BNE   XIT       nope, then leave
               C154 DISTD   EQU   *
C154 BD    CD24             JSR   PCRLF
C157 BD    CD24             JSR   PCRLF
C15A 17    010F             LBSR  DISPLY    yes, print date/time once
C15D BD    CD24             JSR   PCRLF
C160 20    31               BRA   XIT       return to FLEX
```

```
            C162  START1  EQU   *
C162 8E   C2EE            LDX   #OPLMSG
C165 BD   CD1E            JSR   PSTRNG    print intro

            C168  START2  EQU   *
C168 17   0101            LBSR  DISPLY    print time/date

            C16B  START3  EQU   *
C16B BD   CB4E            JSR   STAT      check keyboard status
C16E 26   0A              BNE   START4    brnch if char waiting

C170 17   0125            LBSR  READ
C173 F1   C122            CMPB  XTEMP     check for 1 second roll over
C176 27   F3              BEQ   START3    if the time didn't change

C178 20   EE              BRA   START2

            C17A  START4  EQU   *
C17A BD   CD09            JSR   INCH      get waiting character

C17D 84   5F              ANDA  #$5F      fold lower case to upper
C17F 81   54              CMPA  #'T       set time?
C181 27   13              BEQ   SETIME

C183 81   44              CMPA  #'D       set date?
C185 27   54              BEQ   SEDATE

C187 81   52              CMPA  #'R       return to FLEX?
C189 26   D7              BNE   START1
C18B 20   06              BRA   XIT       return to FLEX
            C18D  ERR1    EQU   *
C18D 8E   C298            LDX   #BADYR    get bad year msg
C190 BD   CD1E            JSR   PSTRNG    print it
            C193  XIT     EQU   *
C193 7E   CD03            JMP   WARMS     return to FLEX
                  *
                  * SET THE TIME
                  *
                  *     The input is in the form of:
                  *
                  *          "HH:MM:SS"
                  *
                  *     where "HH", "MM" and "SS" are 2 digit characters
                  *
                  *     "HH" has a maximum range of 0 to 23
                  *     "MM" and "SS" have a maximum range of 0 to 59
                  *
            C196  SETIME  EQU   *
C196 B6   CC02            LDA   EOL       get TTY end of line char
C199 34   02              PSHS  A         and save it
C19B 7F   CC02            CLR   EOL       set EOL char to null
C19E 8E   C321            LDX   #TIMST
C1A1 BD   CD1E            JSR   PSTRNG
C1A4 BD   CD1B            JSR   INBUF     get time string

C1A7 BD   CD42            JSR   GETHEX    get hours digits

C1AA 1F   10              TFR   X,D
C1AC C1   23              CMPB  #$23
C1AE 22   23              BHI   SETIM1
C1B0 E7   24              STB   HOUR,Y

C1B2 BD   CD42            JSR   GETHEX    get minute digits
C1B5 25   1C              BCS   SETIM1    if illegal value

C1B7 1F   10              TFR   X,D
C1B9 C1   59              CMPB  #$59
C1BB 22   16              BHI   SETIM1
C1BD E7   23              STB   MIN,Y

C1BF BD   CD42            JSR   GETHEX    get second digits
C1C2 25   0F              BCS   SETIM1    if illegal value

C1C4 1F   10              TFR   X,D
C1C6 C1   59              CMPB  #$59
C1C8 22   09              BHI   SETIM1
C1CA E7   22              STB   SEC,Y
```

```
C1CC 35   02              PULS  A         restore EOL char
C1CE B7   CC02            STA   EOL

C1D1 20   8F              BRA   START1
            C1D3  SETIM1  EQU   *
C1D3 8E   C370            LDX   #ILVMSG
C1D6 BD   CD1E            JSR   PSTRNG
C1D9 20   BB              BRA   SETIME    start again
                  *
                  * SET THE DATE
                  *
                  *     The input is in the form of:
                  *
                  *          "DAY,MON,DD[,YYYY]"
                  *
                  *      where "DAY" is a 3 character string for
                  *      the day of the week (SUN - SAT)
                  *
                  *      where "MON" is a 3 character string for
                  *      the month of the year (JAN - DEC)
                  *
                  *      where "DD" is a 2 digit value for the day
                  *      of the month (1 - 31)
                  *
            C1DB  SEDATE  EQU   *
C1DB 8E   C34F            LDX   #DATST
C1DE BD   CD1E            JSR   PSTRNG
C1E1 BD   CD1B            JSR   INBUF

C1E4 30   8D 00CB         LEAX  DOWTBL,PCR
C1E8 8D   44              BSR   SEDAT2    convert day of week string
C1EA 25   39              BCS   SEDAT1    if input error
C1EC E7   25              STB   DOW,Y

C1EE 30   8D 00D7         LEAX  MONTBL,PCR
C1F2 8D   3A              BSR   SEDAT2    convert month string to bin
C1F4 25   2F              BCS   SEDAT1    if illegal value
C1F6 F7   CC0E            STB   SYDR      save in FLEX date reg
C1F9 C1   09              CMPB  #9        convert to BCD
C1FB 23   02              BLS   *+4
C1FD CB   06              ADDB  #6
C1FF E7   27              STB   MON,Y     save it

C201 BD   CD42            JSR   GETHEX    get day of month digit
C204 25   1F              BCS   SEDAT1    if illegal value

C206 1F   10              TFR   X,D
C208 C1   31              CMPB  #$31
C20A 22   19              BHI   SEDAT1
C20C E7   26              STB   DOM,Y
                  *
                  * Convert month to binary and save
                  * in FLEX's date register.
                  *
C20E 34   04              PSHS  B         save hex month
C210 54                   LSRB            get ten's digit
C211 54                   LSRB
C212 54                   LSRB
C213 54                   LSRB
C214 86   0A              LDA   #10       muliply by 10
C216 3D                   MUL
C217 35   02              PULS  A         get back hex month
C219 84   0F              ANDA  %00001111 keep 1's digit
C21B 34   02              PSHS  A         save back
C21D EB   E0              ADDB  0,S+      add on 10's digit
C21F F7   CC0F            STB   SYDR+1    save in FLEX date reg

C222 16   FF3D            LBRA  START1

            C225  SEDAT1  EQU   *
C225 8E   C370            LDX   #ILVMSG
C228 BD   CD1E            JSR   PSTRNG
C22B 16   FF34            LBRA  START1

            C22E  SEDAT2  EQU   *
C22E C6   03              LDB   #3
C230 CE   C122            LDU   #XTEMP
```

```
               C233  SEDAT3  EQU   *
C233 BD   CD27               JSR   NXTCH     get a character
C236 24   08                 BCC   SEDAT4    if alpha-numeric

C238 81   20                 CMPA  #SP
C23A 27   F7                 BEQ   SEDAT3    eat leading blanks
C23C 81   0D                 CMPA  #CR
C23E 27   29                 BEQ   SEDAT7

               C240  SEDAT4  EQU   *
C240 84   5F                 ANDA  #$5F      fold lower case to upper
C242 81   41                 CMPA  #'A       don't use numeric chars
C244 2D   23                 BLT   SEDAT7
C246 A7   C0                 STA   0,U+
C248 5A                      DECB
C249 26   EB                 BNE   SEDAT3

C24B BD   CD27               JSR   NXTCH     eat terminator character
C24E FE   C122               LDU   XTEMP+0   -> get first two chars
C251 B6   C124               LDA   XTEMP+2

C254 C6   01                 LDB   #1        preset counter

               C256  SEDAT5  EQU   *
C256 11A3 84                 CMPU  0,X
C259 26   07                 BNE   SEDAT6    try next string
C25B A1   02                 CMPA  2,X
C25D 26   03                 BNE   SEDAT6    try next string

C25F 1C   FE                 CLC             indicate success
C261 39                      RTS

               C262  SEDAT6  EQU   *
C262 5C                      INCB
C263 30   03                 LEAX  3,X       try next string
C265 6D   84                 TST   0,X       end of table?
C267 26   ED                 BNE   SEDAT5

               C269  SEDAT7  EQU   *
C269 1A   01                 SEC             indicate failure
C26B 39                      RTS
                     *
                     * UPDATE DISPLAY ONCE A SECOND
                     *
               C26C  DISPLY  EQU   *
C26C BD   C387               JSR   ZRDTIM    update time/date string

C26F 86   0D                 LDA   #CR
C271 BD   CD0F               JSR   OUTCH

C274 8E   C102               LDX   #YDATE
C277 AD   9F F80C            JSR   [PDATA]   print date string
C27B 86   20                 LDA   #SP
C27D BD   CD0F               JSR   OUTCH
C280 BD   CD0F               JSR   OUTCH

C283 8E   C112               LDX   #YTIME
C286 AD   9F F80C            JSR   [PDATA]   print time string

C28A 86   20                 LDA   #SP
C28C BD   CD0F               JSR   OUTCH
C28F BD   CD0F               JSR   OUTCH

C292 8D   04                 BSR   READ
C294 F7   C122               STB   XTEMP
C297 39                      RTS
                     *
                     * READ THE SECONDS REGISTER
                     *
C298 E6   22         READ    LDB   SEC,Y
C29A 39                      RTS
                     *
                     * Data Area
                     *
C29B 49 6E 76 61     BADYR   FCC   /Invalid YEAR specified!/
C2B2 04                      FCB   EOT
```

```
               C2B3  DOWTBL  EQU   *
C2B3 53 55 4E                FCC   /SUN/
C2B6 4D 4F 4E                FCC   /MON/
C2B9 54 55 45                FCC   /TUE/
C2BC 57 45 44                FCC   /WED/
C2BF 54 48 55                FCC   /THU/
C2C2 46 52 49                FCC   /FRI/
C2C5 53 41 54                FCC   /SAT/
C2C8 00                      FCB   0         end of table
               C2C9  MONTBL  EQU   *
C2C9 4A 41 4E                FCC   /JAN/
C2CC 46 45 42                FCC   /FEB/
C2CF 4D 41 52                FCC   /MAR/
C2D2 41 50 52                FCC   /APR/
C2D5 4D 41 59                FCC   /MAY/
C2D8 4A 55 4E                FCC   /JUN/
C2DB 4A 55 4C                FCC   /JUL/
C2DE 41 55 47                FCC   /AUG/
C2E1 53 45 50                FCC   /SEP/
C2E4 4F 43 54                FCC   /OCT/
C2E7 4E 4F 56                FCC   /NOV/
C2EA 44 45 43                FCC   /DEC/
C2ED 00                      FCB   0         end of table

C2EE 0D0A            OPLMSG  FDB   CRLF
C2F0 53 65 74 20             FCC   'Set time (T), set date (D), '
C30C 72 65 74 75             FCC   'return to FLEX (R)'
C31E 0D0A                    FDB   CRLF
C320 04                      FCB   EOT

C321 0D0A            TIMST   FDB   CRLF
C323 49 6E 70 75             FCC   'Input time in 24 hour '
C339 66 6F 72 6D             FCC   'format "23:59:59"'
C34A 0D 0A 3E 20             FCB   CR,LF,'>,SP
C34E 04                      FCB   EOT

C34F 0D0A            DATST   FDB   CRLF
C351 49 6E 70 75             FCC   'Input date in "DAY,MON,DD"'
C36B 0D 0A 3E 20             FCB   CR,LF,'>,SP
C36F 04                      FCB   EOT

C370 49 6C 6C 65     ILVMSG  FCC   'Illegal input value!!!'
C386 04                      FCB   EOT
                     *
                     * This routine reads the time and date and stores
                     * the values in YTIME and VDATE respectively
                     *
               0014  STATUS  EQU   20        status register

               C387  RDTIME  EQU   *
C387 34   76                 PSHS  A,B,X,Y,U save the important stuff
C389 108E F700               LDY   #CLOCK    -> start of the MM58167 reg
                     *
                     * Start by reading the time
                     *
C38D CE   C112               LDU   #YTIME    -> where to put time

C390 86   04                 LDA   #HOUR
C392 8D   16                 BSR   RDTIM1    convert & store hours

C394 86   03                 LDA   #MIN
C396 8D   12                 BSR   RDTIM1    convert & store minutes
C398 86   02                 LDA   #SEC
C39A 8D   0E                 BSR   RDTIM1    convert & store seconds

C39C 86   04                 LDA   #EOT
C39E A7   5F                 STA   -1,U      set end of string
C3A0 20   38                 BRA   RDTIM4    now read the data
                     *
                     * Read the specified register
                     *
               C3A2  XREAD   EQU   *
C3A2 E6   A6                 LDB   A,Y
C3A4 6D   A8 14              TST   STATUS,Y  test for register roll over
C3A7 26   F9                 BNE   XREAD
C3A9 39                      RTS
                     *
```

```
                         * Convert BCD digits to ASCII BCD
                         *
                 C3AA    RDTIM1  EQU     *
C3AA 8D    F6                    BSR     XREAD      read the specified register
C3AC 1F    98                    TFR     B,A
                         *
                         * Convert MSB
                         *
C3AE 44                          LSRA
C3AF 44                          LSRA
C3B0 44                          LSRA
C3B1 44                          LSRA
                         *
                         * Convert LSB
                         *
C34A 0D 0A 3E 20                 FCB     CR,LF,'>,SP
C34E 04                          FCB     EOT

C34F 000A                DATST   FDB     CRLF
C351 49 6E 70 75                 FCC     'Input date in "DAY,MON,DD"'
C36B 0D 0A 3E 20                 FCB     CR,LF,'>,SP
C36F 04                          FCB     EOT

C370 49 6C 6C 65         ILVMSG  FCC     'Illegal input value!!!'
C386 04                          FCB     EOT
                         *
                         * This routine reads the time and date and stores
                         * the values in YTIME and YDATE respectively
                         *
                 0014    STATUS  EQU     20         status register

                 C387    RDTIME  EQU     *
C387 34    76                    PSHS    A,B,X,Y,U save the important stuff
C389 108E F700                   LDY     #CLOCK     -> start of the MM58167 reg
                         *
                         * Start by reading the time
                         *
C38D CE    C112                  LDU     #YTIME     -> where to put time

C390 86    04                    LDA     #HOUR
C392 8D    16                    BSR     RDTIM1     convert & store hours

C394 86    03                    LDA     #MIN
C396 8D    12                    BSR     RDTIM1     convert & store minutes
C398 86    02                    LDA     #SEC
C39A 8D    0E                    BSR     RDTIM1     convert & store seconds

C39C 86    04                    LDA     #EOT
C39E A7    5F                    STA     -1,U       set end of string
C3A0 20    38                    BRA     RDTIM4     now read the data
                         *
                         * Read the specified register
                         *
                 C3A2    XREAD   EQU     *
C3A2 E6    A6                    LDB     A,Y
C3A4 6D    A8 14                 TST     STATUS,Y  test for register roll over
C3A7 26    F9                    BNE     XREAD
C3A9 39                          RTS
                         *
                         * Convert BCD digits to ASCII BCD
                         *
                 C3AA    RDTIM1  EQU     *
C3AA 8D    F6                    BSR     XREAD      read the specified register
C3AC 1F    98                    TFR     B,A
                         *
                         * Convert MSB
                         *
C3AE 44                          LSRA
C3AF 44                          LSRA
C3B0 44                          LSRA
C3B1 44                          LSRA
                         *
                         * Convert LSB
                         *
C3B2 C4    0F                    ANDB    #$F
                         *
                         * Convert MSB & LSB to ASCII
```

```
                         *
C3B4 C3    3030                  ADDD    #'0*256+'0
C3B7 ED    C1                    STD     0,U++      store characters

C3B9 86    3A                    LDA     #':
C3BB A7    C0                    STA     0,U+

C3BD 39                          RTS
                         *
                         * Convert binary number to corresponding string
                         *
                 C3BE    RDTIM2  EQU     *
C3BE 8D    E2                    BSR     XREAD      read the specified register
C3C0 C1    09                    CMPB    #9         check BCD range (> 9?)
C3C2 23    02                    BLS     *+4
C3C4 C0    06                    SUBB    #6         make it binary
C3C6 34    04                    PSHS    B
C3C8 58                          ASLB
C3C9 EB    E0                    ADDB    0,S+
C3CB 3A                          ABX                -> pickup string
C3CC C6    03                    LDB     #3         move 3 characters

                 C3CE    RDTIM3  EQU     *
C3CE A6    80                    LDA     0,X+       move table char to ...
C3D0 A7    C0                    STA     0,U+       parameter string area
C3D2 5A                          DECB
C3D3 26    F9                    BNE     RDTIM3

C3D5 86    20                    LDA     #SP
C3D7 A7    C0                    STA     0,U+       install separator

C3D9 39                          RTS
                         *
                         * Read the data
                         *
                 C3DA    RDTIM4  EQU     *
C3DA CE    C102                  LDU     #YDATE     -> where to put date

C3DD 30    8D FDCF               LEAX    DOWTBL-3,PCR -> day of week table
C3E1 86    05                    LDA     #DOW       read day of week
C3E3 8D    D9                    BSR     RDTIM2     convert to ascii string

C3E5 30    8D FEDD               LEAX    MONTBL-3,PCR -> month table
C3E9 86    07                    LDA     #MON       read month
C3EB 8D    D1                    BSR     RDTIM2     convert to ascii string

C3ED 86    06                    LDA     #DOM       read day of month
C3EF 8D    B9                    BSR     RDTIM1     convert to ascii BCD

C3F1 86    20                    LDA     #SP
C3F3 A7    5F                    STA     -1,U
                         *
                         * The following code stores the year portion of
                         * the date in the YDATE string. The last 2
                         * digits in the year are gotten from the
                         * FLEX date register and converted to a 2
                         * digit ascii value.
                         *
C3F5 CC    3139.                 LDD     #'1*256+'9 "19"
C3F8 ED    C1                    STD     0,U++
C3FA 86    CC10                  LDA     SYDR+2     get binary year
C3FD 8D    2C                    BSR     BINASC     convert to ascii
C3FF ED    C1                    STD     0,U++

C401 86    04                    LDA     #EOT
C403 A7    C4                    STA     0,U        set end of string

C405 35    76                    PULS    A,B,X,Y,U restore the important stuff
C407 39                          RTS
                         *
                         *  ASCBIN - this routine converts 2 ascii chars
                         *           to binary
                         *
                         *      entry: ACC D contains 2 ascii characters
                         *
                         *      exit:  ACC A contains binary equivalent
                         *             Carry is clear if digits are
                         *          valid decimal digits (0-9),
                         *          otherwise carry is set
                         *
                         *      accumulators A and B are used and not
                         *        restored.
                         *
                   C408  ASCBIN  EQU   *
C408 81   30                     CMPA  #'0       make sure first ascii
C40A 25   1C                     BLO   BADIG     char is between
C40C 81   39                     CMPA  #'9       0 and 9
C40E 22   18                     BHI   BADIG

C410 C1   30                     CMPB  #'0       make sure second ascii
C412 25   14                     BLO   BADIG     char is between
C414 C1   39                     CMPB  #'9       0 and 9
C416 22   10                     BHI   BADIG

C418 84   0F                     ANDA  #%00001111 keep low 4 bits
C41A C4   0F                     ANDB  #%00001111
C41C 34   04                     PSHS  B         save 2nd digit
C41E C6   0A                     LDB   #10
C420 3D                          MUL             multiply first by 10
C421 1F   98                     TFR   B,A       B -> A
C423 AB   E0                     ADDA  0,S+      add in 1's digit
C425 1C   FE                     CLC             set good RC
C427 39                          RTS             return
C428 1A   01             BADIG   SEC             set bad RC
C42A 39                          RTS             return
                         *
                         *  BINASC - this routine converts a 1
                         *          byte binary number (<= 99 base 10)
                         *          to ascii.
                         *
                         *    entry: ACC A contains binary number
                         *    exit:  ACC D contains 2 digit ascii rep
                         *
                         *    Accumulators A and B are used and not
                         *      restored.
                         *
                   C42B  BINASC  EQU   *
C42B 34   02                     PSHS  A         save binary #

C42D C6   08                     LDB   #8        # bits to shift out
C42F 4F                          CLRA            hold BCD value here

                   C430  DOBLE   EQU   *
                         *
                         * Double current BCD result before
                         * shifting out a bit from the binary
                         * number.
                         *
C430 34   02                     PSHS  A         double BCD value
C432 AB   E0                     ADDA  0,S+

C434 19                          DAA             (in BCD)
C435 68   E4                     LSL   0,S       shift out a bit
C437 24   07                     BCC   CHK       branch if bit=0
C439 34   02                     PSHS  A         add 1 to current
C43B 86   01                     LDA   #1        BCD value
C43D AB   E0                     ADDA  0,S+
C43F 19                          DAA             (in BCD of course)

C440 5A                  CHK     DECB            done yet?
C441 26   ED                     BNE   DOBLE     no, then continue

C443 32   61                     LEAS  1,S       clean up stack
                         *
                         * Convert BCD # in A to ascii
                         *
C445 34   02                     PSHS  A         save BCD value
C447 44                          LSRA            cvt 10's dig to ascii
C448 44                          LSRA
C449 44                          LSRA
C44A 44                          LSRA
C44B 8A   30                     ORA   #'0
C44D 35   04                     PULS  B         cvt 1's dig to ascii
C44F C4   0F                     ANDB  #%00001111
C451 CA   30                     ORB   #'0
C453 39                          RTS             return
                                 END   START

0 ERROR(S) DETECTED


SYMBOL TABLE:

ADDBX  CD36   ASCBIN C408   ASREAD 0001   ASWRIT 0002   BAC    0008
BADIG  C428   BADYR  C29B   BAK    0005   BAS    0003   BELL   0007
BIN    0000   BINASC C42B   BS     CC00   BSE    CC07   BUFPNT CC14
CHK    C440   CLASS  CD21   CLN    CC1A   CLOCK  F700   CMD    0002
CMDFLG CC28   COC    CC29   COLDS  CD00   CR     000D   CRLF   000A
CURC   CC1B   DAT    0007   DATST  C34F   DDRV   DE00   DEL    CC01
DEPTH  CC03   DIR    0009   DIRTS  0005   DISPLY C26C   DISTD  C154
DOBLE  C430   DOCMND CD4B   DOM    0006   DOS    CC00   DOW    0005
DOWTBL C2B3   DPLMSG C2EE   EJECT  CC08   ENV    CC20   EOL    CC02
EOT    0004   ERR1   C1BD   ESC    CC0A   ESCRR  CC16   FACP   0010
FADP   0040   FAMP   0020   FAWP   0080   FCBBAS 0002   FCBASE D409
FCBCDA 002F   FCBCP  001E   FCBCRN 0020   FCBCUR D40B   FCBDI  0022
FCBDN  0003   FCBEDA 0013   FCBESB 0001   FCBFA  000F   FCBFC  0000
FCBFCD 0019   FCBFDD 0032   FCBFS  0015   FCBFSM 0017   FCBLEN 0140
FCBLP  001C   FCBNAM 0004   FCBNMB 0024   FCBR1  0023   FCBRS1 0010
FCBRS2 0018   FCBSB  0040   FCBSCF 003B   FCBSDR 0035   FCBSDA 0011
FCBVER D435   FCDDAY 001A   FCDMTH 0019   FCDYR  001B   FIA    CC26
FIEF   CC2F   FLEX   CD00   FMS    D406   FMSCAL D406   FMSCLS D403
FMSERR CC20   FMSINT D400   FOA    CC24   FSMRAN 0002   FSMSEQ 0000
GETCHR CD15   GETFIL CD2D   GETHEX CD42   HOUR   0004   ILVMSG C370
INBUF  CD1B   INCH   CD09   INCH2  CD0C   INDEC  CD48   IOFLG  CC21
ISWTCH CC23   LAD    CC1B   LF     000A   LNEBUF C080   LOAD   CD30
LSTRM  CC11   MAP    CC00   MEMEND CC2B   MIN    0003   MON    0007
MONTBL C2C9   NULL   CC05   NXTCH  CD27   OSWTCH CC22   OUT    0008
OUTADR CD45   OUTCH  CD0F   OUTCH2 CD12   OUTDEC CD39   OUTHEX CD3C
PAU    CC09   PCRLF  CD24   PDATA  F80C   POUT   C0E4   PRCHK  CC08
PREVC  CC19   PRINIT CCC0   PRT    000A   PSTRNG CD1E   PUTCHR CD18
RDTIM1 C3AA   RDTIM2 C3BE   RDTIM3 C3CE   RDTIM4 C3DA   RDTIME C387
READ   C298   RENTER CD06   RPTERR CD3F   RSTRIO CD2A   SBDATA 0044
SBLINK 0040   SBRS1  0042   SCFMSC 006F   SCFSC  0000   SCR    0006
SEC    0002   SEDAT1 C225   SEDAT2 C22E   SEDAT3 C233   SEDAT4 C240
SEDAT5 C256   SEDAT6 C262   SEDAT7 C269   SEDATE C108   SETEXT CD33
SETIM1 C1D3   SETIME C196   SFA    C980   SIRBRE 0023   SIRDAY 0024
SIRFSB 001D   SIRFSE 001F   SIRFSS 0021   SIRLEN 0028   SIRMTH 0023
SIRMTS 0026   SIRNAM 0010   SIRTS  0003   SIRVOL 0018   SIRYR  0025
SP     0020   SPS    C700   START  C125   START1 C162   START2 C168
START3 C16B   START4 C17A   STAT   CD4E   STATUS 0014   STKA   C000
SYDR   CC0E   SYDRV  CC0B   SYS    0004   SYSCON CC4E   SYSCR1 CC0D
SYSCR2 CC2A   SYSCR3 CC30   SYSCR4 CCFB   SYSFCB D840   TAB    CC06
TIMST  C321   TRADDR CC1E   TRFLG  CC1D   TXT    0001   UCA    C100
UCTA   CC12   URAM   0000   VN     0002   WARMS  CD03   WIDTH  CC04
WKDRV  CC0C   XBOR   0016   XCLOSE 0004   XDELET 000C   XFND   0014
XGIR   0007   XGRB   0011   XIT    C193   XNSS   000F   XODIR  0006
XOREAD 0001   XOSIR  0010   XOUPDT 0003   XOWRIT 0002   XPIR   0008
XPOSN  0015   XPRB   0012   XREAD  C3A2   XRENAM 000D   XRES1  0008
XRES2  000E   XRES3  0013   XREWND 0005   XRSS   0009   XRWNB  0000
XTEMP  C122   XWSG   000A   YDATE  C102   YTIME  C112   ZROTIM C387
```

Classified Advertising

TELETYPE Model 43 PRINTER - with serial (RS232) interface, and full ASCII keyboard. **LIKE NEW** - New cost $1295.00 - ONLY **$759.00** ready to run - Call Tom - Larry - Bob, CPI 615 842-4600

***

For Sale: Motorola 128K Memory Boards, removed from SWPTC S/09 $795.00, SWPTC 8212 Terminals Demostrators $795.00, Hazelwood Dynamic 64K Memory Boards $395.00 Call ask for Tom 615/ 842-4600

***

SWTPC 6809 system, 56K, MPL2, MPT, MPS2, 2 DSOD Drives, software. $599.
Tom Harmon, 15418 Diana Ln., ouston, TX 77062
713 480-6075.

***

OS-9™ SOFTWARE FOR COCO

SDISK — Standard disk driver module allows the use of 35, 40, or 80 track single- and double-sided drives with CoCo OS-9 plus you gain the ability to read/write/format the standard OS-9 single- and double-density disk formats used on other OS-9 systems. **$29.95**

BOOTFIX — To make bootable double-sided disks (requires double-sided drive and SDISK). **$9.95**

SDISK + BOOTFIX — When ordered together. **$35.95**

FILTER KIT #1 — Eleven utilities used as filters (with pipes) to give you "wild card" directory lists, copies, deletes, moves, lists, pagination, etc. **$29.95**

FILTER KIT #2 — Command Macro Generator to build new commands by combining old ones,and 9 other utilities. **$29.95**

HACKER'S KIT #1 — Disassembler and memory dump/fill utilities allow you to disassemble OS-9 assembly code from disk or memory. **$24.95**

Send SASE for current catalog.
Terms: Prepaid check, MO, Visa, Mastercard or COD. Add $1 S&H, (COD $3 extra).

D.P. Johnson, 7655 S.W. Cedarcrest St.
Portland, OR 97223 (503) 244-8152
(We appreciate your calling only between 9-11 AM PST.)

OS-9 is a trademark of Microware and Motorola Inc.

SWTP S/09 COMPUTER
128K, CT-82 TERMINAL, DUAL 8" FLOPPIES, MP-WP INTERFACE. JUST LIKE NEW. SOFTWARE INCLUDED.
$3400.00 CALL LIEF days:(512)454-1215
nights:(512)258-5751

***

FOR SALE: PR-40 printer $60, APTEK 4K 1702 SS50 EPROM board, F&D 1702 programmer (uses MPLA) both $100, Universal Data 212/103 modem $300, DS68 103 modem card populated, never used, with EDC coupler $50.
Gordon (504)889-1224

***

SELLING OUT! Gimix #39 Mainframe w/dual 8" Qume Floppy Drives, 340K Gimix static ram, Gimix intelligent I/O Processor board, other serial/parallel I/O boards, PRIVAC hi-res graphics, META-LABS Z80 CPM board, Windrush Prom Burner, 5 Meg HHH Hard Disk, Z19 terminal, 9511. Much software. Original Cost over $13000. All for only $6800. Will sell some pieces separately.
Call Dick evenings at 215-257-3992.

***

FOR SALE: The following SWTP-6809 Flex/Uniflex components: 2-S/09+ mainframes, 2-8212's, 3-64K boards, 2- dual 8-inch diskette drive, QUME 45cps printer with tractor and sheetfeeder and MP-QP interface board. MUST SELL.
Make offe s to Richard Davidson, (517)332-5989.

***

ALL GIMIX 6809 SYSTEM--CLASSY CHASSIS, 56 KB STATIC RAM, DUAL 40 TRACK DSDD DRIVES (TANDON), DMA FLOPPY DISK CONTROLLER, 2 SERIAL CARDS (4 PORTS), 2 PARALLEL CARDS (4 PORTS), GIMIX CABLING AND CONNECTORS, FLEX 9.0, OS-9 LEVEL 1, TSC BASIC, TSC DIAGNOSTICS, TSC ASSEMBLER, TSC EDITOR, TSC UTILITIES, STYLO AND MOUNTAINS OF OTHER APPLICATIONS SOFTWARE. GUARANTEED NEW CONDITION, NO GLITCHES OR INTERMITTENTS. MUST SELL, BEST OFFER OVER $1000 PLUS SHIPPING.
CALL (505) 662-7417 NITES. TOM GARDINER, 393 EL VIENTO, LOS ALAMOS, NM, 87544.

***

COMPILER EVALUATION SERVICES
By: Ron Anderson

The S.E. MEDIA Division of Computer Publishing Inc., Is offering the following 'SUBSCRIBER SERVICE':

COMPILER COMPARISION AND EVALUATION REPORT

Due to the constant and rapid updating and enhanchment of numerous compilers, and the different utility, appeal, speed, level of communication, memory usage, etc., of different compilers, the following services are now being offered with periodic updates.

This service, with updates, will allow you who are wary or confused by the various claims of compiler vendors, an opportunity to review comparisons, comments, benchmarks, etc., concerning the many different compilers on the market, for the 6809 microcomputer. Thus the savings could far offset the small cost of this service.

Many have purchased compilers and then discovered that the particular compiler purchased either is not the most efficient for their purposes or does not contain features necessary for their application. Thus the added expense of purchasing additional compiler(s) or not being able to fully utilize the advantages of high level language compilers becomes too expensive.

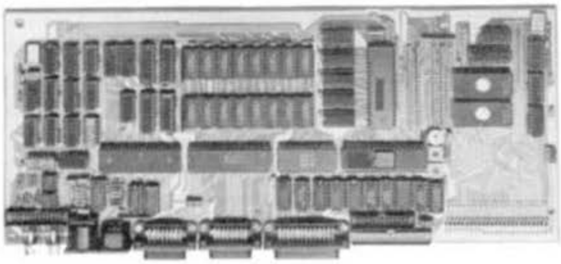
The following COMPILERS are reviewed initially, more will be reviewed, compared and benchmarked as they become available to the author:

PASCAL "C" GSPL WHIMISCAL PL/9

Initial Subscription - $39.95
(Includes 1 year updates)
Updates for 1 year - $14.50

S.E. MEDIA - CPI
5900 Cassandra Smith, POB 794
Hixson, TN 37343
615 842-4601

MICROKEY 4500 A PROFESSIONAL 6809/ FLEX COMPUTER SYSTEM



TECHNICAL EXCELLENCE IN A SINGLE-BOARD COMPUTER FOR SYSTEMS DEVELOPMENT, O.E.M., OR APPLICATIONS.

BY USING THE SAME BOARD IN YOUR TARGET SYSTEM AS YOUR DEVELOPMENT SYSTEM, YOU CAN SAVE TIME AND MONEY!

µKEY Features

- 6809/2MHz PROCESSOR WITH 128K RAM AS STANDARD.
- FLEX OPERATING SYSTEM OR polyFORTH AVAILABLE.
- 3½ and 5¼ in. DISK DRIVES CAN BE MIXED.
- 16 COLOR, HIGH RESOLUTION GRAPHICS.
- NORMAL, ULTRA-HIGH RES. AND PAGE FORMAT MONOCHROME MODES.
- TWO INDEPENDENT VIDEO OUTPUTS.
- MULTI-TASKING
- INTERLACE MODE.

SPECIAL OFFER FOR LIMITED PERIOD

DEV SYSTEM TRIPLE DRIVE	**$1,899**
TARGET BOARD 128K RAM	**$450**

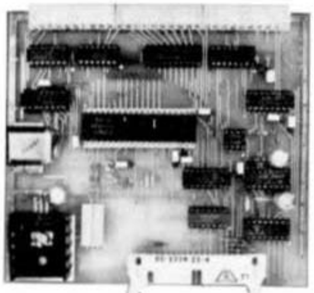
SHIPPING AND TAX EXTRA

ORDERS AND ENQUIRIES TO:
MICR●KEY LTD., 98a, St. James's Street,
Brighton, Sussex, England. Tel: 0273-672911

FLEX is a trademark of Tech. Sys. Consultants
polyFORTH is a trademark of FORTH INC.

OS-9 SUPPORT FOR FD-2

NEW!



NEW!

Run double density on any S-50 6800 or 6809 computer. Who else can offer this capability at these low prices? The FD-2 features:

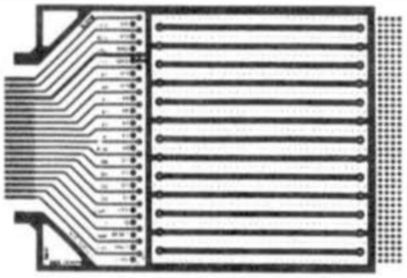
- Control of up to four 5¼" DS/DD Drives
- SS-30 or SS-30C compatible
- Use Flex, OS-9, or Star Dos operating systems
- 2.0 MHZ operation with no "slow I/O" required
- Compatible with SWTPC DC1, DC2, DC3, or DC4 controllers

FD-2	Assembled/Tested Controller Card	$149.95
DRV-68	6800 double density drivers + format program	$ 19.95
DRV-69	6809 double density drivers + format program	$ 29.95
DRV-09	FD 2 Disk Drivers for OS-9 (Source)	$100.00
STAR-DOS	For SWTPC & FD-2	$ 75.00

PERIPHERAL TECHNOLOGY

"Supplying Floppy Disk Controllers Since 1978"
3760 Lower Roswell Road
Marietta, Georgia 30067
VISA/MASTERCARD/CHECK/COD 404/973-0042

™OS-9 is a trademark of Microware and Motorola.



XPNDR1™

The XPNDR1 card is designed to plug into the CoCo 40 pin extension connector where power, the 6809E address and data buses, clocks and control signals are available for your prototype hardware. You can add connectors for ROM paks, put together a communications interface, add peripheral chips to support graphics and voice for simulations or game development, build circuits to monitor and control external events, participate in educational projects, and much, much more.

Shown is the bottom side of the card where the CoCo signals are identified for fast, accurate wiring, and you can use the extensive power and ground buses.

Made from double-sided glass/epoxy with plated thru-holes, the 4.3 x 6.3 inch XPNDR1 is drilled for standard 0.3 and 0.6 inch wide dual in-line IC sockets. Plus on the outboard end is a grid of solder pads on 0.1 inch centers. The edge connector and grounding tabs are gold plated.

This is a tough card with plenty of clearance between oversize traces and pads; it will stand up to a lot of prototyping.

$18.95 each or 2 for $36

Includes 8 pages of *Application Notes* to help you learn about chips and how to connect them to your CoCo.

SuperGuide™

Here is a precision card guide made (of teflon) to align and support prototype cards in the CoCo ROM pak slot. A real breakthrough problem solver. Insert or remove instantly. Order one for your XPNDR1 cards.

$3.95 each

For immediate shipment (and we'll pay First Class postage!) send check or money order, or the number and expiration date of your VISA or MASTERCARD to:

ROBOTIC MICROSYSTEMS
BOX 30807 SEATTLE, WA 98103

GOOD NEWS!

C for the 6809 WAS NEVER BETTER!

INTROL-C/6809,Version 1.5

Introl's highly acclaimed 6809 C compilers and cross-compilers are now more powerful than ever!

We've incorporated a totally new 6809 Relocating Assembler, Linker and Loader. Initializer support has been added, leaving only bitfield-type structure members and doubles lacking from a 100% full K&R implementation. The Runtime Library has been expanded and the Library Manager is even more versatile and convenient to use. Best of all, compiled code is just as compact and fast-executing as ever - and even a bit more so! A compatible macro assembler, as well as source for the full Runtime Library, are available as extra-cost options.

Resident compilers are available under **Uniflex, Flex** and **OS9.**

Cross-compilers are available for **PDP-11/UNIX** and **IBM PC/PC DOS** hosts.

Trademarks:
Introl-C, Introl Corporation
Flex and Uniflex, Technical Systems Consultants
OS9, Microware Systems
PDP-11, Digital Equipment Corp.
UNIX, Bell Laboratories
IBM PC, International Business Machines

For further information, please call or write.

INTROL CORPORATION

647 W. Virginia St.
Milwaukee, WI 53204
(414) 276-2937

FEATURES THE POWERFUL, THIRD GENERATION, MOTOROLA 6809 PROCESSOR!

THE 6809 "UNIBOARD"™

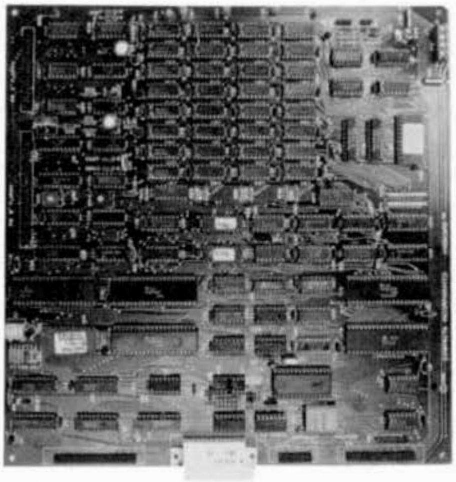
SINGLE BOARD COMPUTER KIT

PERFECT FOR COLLEGES, OEM'S, INDUSTRIAL AND SCIENTIFIC USES!

64K RAM! DOUBLE DENSITY FLOPPY DISK CONTROLLER!

New!

Lower Price!



BLANK PC BOARD

$99.95

WITH PAL'S, AND TWO EPROMS.

FOR 5-1/4 OR 8 INCH SOURCE DISKETTE ADD $10.

$249.00

COMPLETE KIT! FULLY SOCKETED.

PRICE CUT!!

THE COMPACTA UNIBOARD™: Through special arrangement with COMPACTA INC., we are proud to have been selected the exclusive U.S. Mfg. of their new 6809 UNIBOARD™ COMPUTER KIT. Many software professionals feel that the 6809 features probably the most powerful instruction set available today on ANY 8 bit micro. Now, at last, all of that immense computing power is available at a truly unbelievably low price.

ALL SALES ARE MADE SUBJECT TO THE TERMS OF OUR 90 DAY LIMITED WARRANTY. A FREE COPY IS AVAILABLE UPON REQUEST.

FEATURES:

- ★ 64K RAM using 4116 RAMS.
- ★ 6809E Motorola CPU.
- ★ Double Density Floppy Disk Controller for either 5-1/4 or 8 inch drives. Uses WD1793.
- ★ On board 80 x 24 video for a low cost console. Uses 2716 Char. Gen. Programmable Formats. Uses 6845 CRT Controller.
- ★ ASCII keyboard parallel input interface. (6522)
- ★ Serial I/O (6551) for RS232C or 20 MA loop.
- ★ Centronics compatible parallel printer interface. (6522)
- ★ Buss expansion interface with DMA channel. (6844)
- ★ Dual timer for real time clock application.
- ★ Powerful on board system monitor (2732). Features commands such as Go To, Alter, Fill, Move, Display, or Test Memory. Also Read and Write Sectors. Boot Normal, Unknown, and General Flex™.

YOUR CHOICE OF POPULAR DISK OPERATING SYSTEMS:

FLEX™ from TSC	$99
OS9™ from Microware	$199

Specify 5-1/4 or 8 Inch

PC BOARD IS DOUBLE SIDED, PLATED THRU SOLDER MASKED, 11 x 11-1/2 IN.

Digital Research Computers
(OF TEXAS)
P.O. BOX 461565 • GARLAND, TEXAS 75046 • (214) 225-2309

TERMS: Shipments will be made approximately 3 to 6 weeks after we receive your order. VISA, MC, cash accepted. Add $4.00 shipping.
USA AND CANADA ONLY

64K SS-50 STATIC RAM

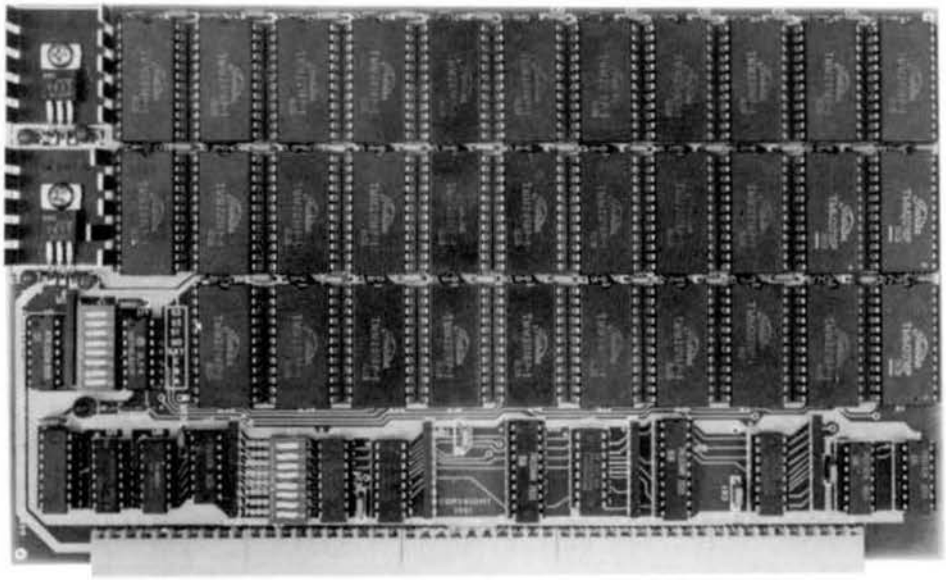
PRICE CUT!!

$149.00 (48K KIT)

NEW!

LOW POWER!

RAM OR EPROM!



BLANK PC BOARD WITH DOCUMENTATION $45

SUPPORT ICs + CAPS -	$18.00
FULL SOCKET SET -	$15.00

ASSEMBLED AND TESTED **ADD $50**

FEATURES:

56K	$169
64K	$199

- ★ Uses new 2K x 8 (TMM 2016 or HM 6116) RAMs.
- ★ Fully supports Extended Addressing.
- ★ 64K draws only approximately 500 MA.
- ★ 200 NS RAMs are standard. (TOSHIBA makes TMM 2016s as fast as 100 NS. FOR YOUR HIGH SPEED APPLICATIONS.)
- ★ Board is configured as 3-16K blocks and 8-2K blocks (within any 64K block) for maximum flexibility.
- ★ 2716 EPROMs may be installed anywhere on Board.
- ★ Top 16K may be disabled in 2K blocks to avoid any I/O conflicts.
- ★ One Board supports both RAM and EPROM.
- ★ RAM supports 2MHZ operation at no extra charge!
- ★ Board may be partially populated in 16K increments.

CLOSE OUT SPECIAL
WE HAVE DROPPED OUR 32K SS-50 STATIC RAM BOARD WHICH USED 2114 LOW POWER RAMS. WE WILL SELL THE REMAINING STOCK OF BLANK PCB'S WITH DATA FOR $17.50 EA. THESE FORMERLY SOLD FOR $50.

16K STATIC RAMS?

The new 2K x 8, 24 PIN, static RAMs are the next generation of high density, high speed, low power. RAMs. Pioneered by such companies as HITACHI and TOSHIBA, and soon to be second sourced by most major U.S. manufacturers, these ultra low power parts, feature 2716 compatible pin out. Thus fully interchangeable ROM/RAM boards are at last a reality, and you get BLINDING speed and LOW power thrown in for virtually nothing.

Digital Research Computers
(OF TEXAS)
P.O. BOX 461565 • GARLAND, TEXAS 75046 • (214) 225-2309

TERMS: Add $2.00 postage. We pay balance. Order under $15 add 75¢ handling. No. C.O.D. We accept Visa and MasterCharge. Tex. Res. add 5% Tax. Foreign orders (except Canada) add 20% P & H. Orders over $50, add 85¢ for insurance.

WINDRUSH MICRO SYSTEMS

UPROM II

PROGRAMS and VERIFIES: I2758, T2508, I2716, T2516, I2732/2732a, MCM68764/6, I2764/2764A, T2564, I27128/27128A, and I27256. I=Intel, T=Texas, M=Motorola.

NO PERSONALITY MODULES REQUIRED!

TRI-VOLT EPROMS ARE NOT SUPPORTED

INTEL's intelligent programming (tm) implemented for Intel 2764, 27128 and 27256 devices. Intelligent programming reduces the average programming time of a 2764 from 7 minutes to 1 minute 15 seconds (under FLEX) with greatly improved reliability.

Fully enclosed pod with 5' of flat ribbon cable for connection to the host computer MC6821 PIA interface board.

MC6809 software for FLEX and OS9 (Level 1 or 2, Version 1.2).

BINARY DISK FILE offset loader supplied with FLEX, MDOS and OS9.

Menu driven software provides the following facilities:

a. FILL .... a selected area of the buffer with a HEX char.
b. MOVE .... blocks of data.
c. DUMP .... the buffer in HEX and ASCII.
d. FIND .... a string of bytes in the buffer.
e. EXAMINE/CHANGE . the contents of the buffer.
f. CRC .... checksum a selected area of the buffer.
g. COPY .... a selected area of an EPROM into the buffer.
h. VERIFY .... a selected area of an EPROM against the buffer.
i. PROGRAM .... a selected area of an EPROM with data in the buffer.
j. SELECT .... a new EPROM type (return to types menu).
k. ENTER .... the system monitor.
l RETURN .... to the operating system.
L. EXECUTE .... any DOS utility (only in FLEX and OS9 versions).

FLEX AND OS9 VERSIONS AVAILABLE FROM GIMIX.   SSB/MDOS CONTACT US DIRECT.

PL/9

- Friendly inter-active environment where you have INSTANT access to the Editor, the Compiler, and the Trace-Debugger, which, amongst other things, can single step the program a SOURCE line at a time. You also have direct access to any FLEX utility and your system monitor.
- 575+ page manual organized as a tutorial with plenty of examples.
- Fast SINGLE PASS compiler produces 8k of COMPACT and FAST 6809 machine code output per minute with no run-time overheads or license fees.
- Fully compatible with TSC text editor format disk files.
- Signed and unsigned BYTES and INTEGERS, 32-bit floating point REALs.
- Vectors (single dimension arrays) and pointers are supported.
- Mathematical expressions: (+), (-), (*), (/), modulus (\), negation (-)
- Expression evaluators: (=), (<>), (<), (>), (>=), (<=)
- Bit operators: (AND), (OR), (EOR/XOR), (NOT), (SHIFT), (SWAP)
- Logical Operators: (.AND), (.OR), (.EOR/XOR)
- Control statements: IF..THEN..ELSE, IF..CASE1..CASE2..ELSE, BEGIN..END, WHILE.., REPEAT..UNTIL, REPEAT..FOREVER, CALL, JUMP, RETURN, BREAK, GOTO.
- Direct access to (ACCA), (ACCB), (ACCD), (XREG), (CCR) and (STACK).
- FULLY supports the MC6809 RESET, NMI, FIRQ, IRQ, SWI, SWI2, and SWI3 vectors. Writing a self-starting (from power-up) program that uses ANY, or ALL, of the MC6809 interrupts is an absolute snap!
- Machine code may be embedded in the program via the 'GEN' statement. This enables you to code critical routines in assembly language and embed them in the PL/9 program (see 'MACE' for details).
- Procedures may be passed and may return variables. This makes them functions which behave as though they were an integral part of PL/9.
- Several fully documented library procedure modules are supplied: IOSUBS, BITIO, HARDIO, HEXIO, FLEXIO, SCIPACK, STRSUBS, BASTRING, and REALCON.

'... THIS IS THE MOST EFFICIENT COMPILER I HAVE FOUND TO DATE.'

Quoted from Ron Andersons FLEX User Notes column in '68. Need we say more?

WORSTEAD LABORATORIES, NORTH WALSHAM,
NORFOLK, ENGLAND.   NR28 9SA.

TEL: 44 (692) 404086
TLX: 975548 WMICRO G

MACE/XMACE/ASM05

All of these products feature a highly productive environment where the editor and the assembler reside in memory together. Gone are the dale of tedius disk load and save operations while you are debugging your code.

- Friendly inter-active environment where you have instant access to the Editor and the Assembler, FLEX utilities and your system monitor.
- MACE can also produce ASMPROCs (GEN statements) for PL/9 with the assembly language source passed to the output as comments.
- XMACE is a cross assembler for the 6800/1/2/3/8 and supports the extended menmonics of the 6303.
- ASM05 is a cross assembler for the 6805.

D-BUG

LOOKING for a single step tracer and mini in-line disassembler that is easy to use?? Look no further, you have found it. This package is ideal for those small assembly language program debugging sessions. D-BUG occupies less than 6K (including its stack and variables) and may be loaded anywhere in memory. All you do is LOAD IT, AIM IT and GO! (80 col VDUs only).

McCOSH 'C'

This is as complete a 'C' compiler as you will find on any operating system for the 6809. It is completely compatible with UNIX VII and only lacks 'bit-fields' (which are of little practical use in an 8-bit world!).

- Produces very efficent assembly language source output with the 'C' source optionally interleaved as comments.
- Built-in optimizer will shorten object code by about 11%.
- Supports interleaved assembly language programs.
- INCLUDES its own assembler. The TSC relocating assembler is only required if you want to generate your own libraries.
- The pre-processor, compiler, optimiser, assembler and loader all run independantly or under the 'CC' executive. 'CC' makes compiling a program to executable object as simple as typing in 'CC,HELLO.C <RETURN>'.

IEEE-488

- SUPPORTS ALL PRINCIPAL MODES OF THE IEEE-488 (1975/8) BUS SPECIFICATION:
  - Talker
  - Listener
  - System Controller
  - Serial Poll
  - Parallel Poll
  - Group Trigger
  - Single or Dual Primary Address
  - Secondary Address
  - Talk only ... Listen only
- Fully documented with a complete reprint of the KILOBAUD article on the IEEE bus and the Motorola publication 'Getting aboard the IEEE Bus'.
- Low level assembly language drivers suitable for 6800, 6801, 6802, 6803, 6808 and 6809 are supplied in the form of listings. A complete back to back test program is also supplied in the form of a listing. These drivers have been extensively tested and are GUARANTEED to work.
- Single S-30 board (4, 8 or 16 addresses per port), fully socketed, gold plated bus connectors and IEEE interface cable assembly.

PRICES

D-BUG	(6809 FLEX only)	$ 75.00
MACE	(6809 FLEX only)	$ 75.00
XMACE	(6809 FLEX only)	$ 98.00
ASM05	(6809 FLEX only)	$ 98.00
PL/9	(6809 FLEX only)	$198.00
'C'	(6809 FLEX only)	$295.00
IEEE-488	with IEEE-488 cable assembly	$298.00
UPROM-II/U	with one version of software (no cable or interface) ..	$395.00
UPROM-II/C	as above but complete with cable and S-30 interface	$545.00
CABLE	5' twist-n-flat 50 way cable with IDC connectors	$ 35.00
S-30 INT	SS-30 interface for UPROM-II	$130.00
EXOR INT	Motorola EXORbus (EXORciser) interface for UPROM-II ...	$195.00
UPROM SFT	Software drivers for 2nd operating system. Specify FLEX or OS9 AND disk size!	$ 35.00
UPROM SRC	Assembly language source (contact us direct)	

ALL PRICES INCLUDE AIR MAIL POSTAGE

Terms: CWO. Payment by Int'l Money Order. VISA or MASTER-CARD also accepted.

WE STOCK THE FOLLOWING COMPANIES PRODUCTS:
GIMIX, SSB, FHL, MICROWARE, TSC, LUCIDATA, LLOYD I/O,
& ALFORD & ASSOCIATES.

FLEX (tm) is a trademark of Technical Systems Consultants, OS-9 (tm) is a trademark of Microware Systems Corporation, MDOS (tm) and EXORciser (tm) are trademarks of Motorola Incorporated.

DYNACALC®

CoCo's Best & Fastest Spreadsheet System

ACCLAIMED BY THE EXPERTS

"DYNACALC is my choice for a CoCo spreadsheet."
Dan Downard, RAINBOW, September, 1984.

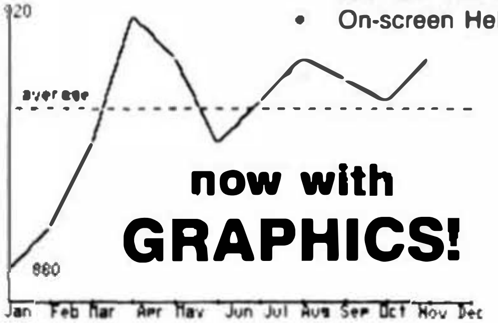
"Eat your heart out, Lotus 1-2-3!"
Scott Norman, HOT CoCo, October, 1984.

NOW ONLY $99⁹⁵

Built-in Features:

- 51 x 24 Display with Lower Case
- Super-fast Smart Screen Refresh
- Auto-Repeat Keyboard Driver
- Keysaver (TM) repeats last command x times
- Disk Operating System (works just like ROM DOS)
- Easy communication with BASIC/DOS programs
- Fast 16-Digit Arithmetic with Scientific Functions
- Summation, Mean, and Standard Deviation Functions
- Logical Functions with String & Numeric Comparison
- String locate command to navigate large worksheets
- Sort full or partial worksheet by columns or rows
- Line, Bar, Hi/Lo/Close, Circle Graphs
- Full Graphics captioning and overlay facility
- Graphics Drivers for all popular Printers
- Joystick/Mouse Driver for Cursor Movement
- Works with any ROM versions, even JDOS
- 33k Available Worksheet Space
- Up to 256 Columns or 256 Rows
- Can use VisiCalc worksheets & training materials
- On-screen Help Messages

RAINBOW
CERTIFICATION
SEAL



now with
GRAPHICS!

FOR 64K DISK SYSTEMS
available from

COMPUTER SYSTEMS CENTER
13461 Olive Blvd.
Chesterfield, MO 63017 USA
(314) 576-5020

CSC

10 to 6 Mon.-Fri.

or your local DYNACALC dealer

Price US$99.95
Outside North America add $10 postage

CANADA- CDN$129.95
RGS MICRO INC.
759, VICTORIA SQUARE 405
MONTREAL H2Y 2J3
TEL.: (514) 287-1563
ORDER LINE ONLY ★ ★ ★
QUEBEC-ONTARIO-MARITIMES
800-361-5338
WESTERN CANADA 800-361-5155

MasterCard
VISA

DYNACALC Reg. U. S. Pat. Off.
Lotus and 1-2-3 are trademarks of Lotus Development Corp.
VisiCalc is a trademark of VisiCorp

'68' MICRO JOURNAL

★ The only ALL 6800 Computer Magazine.

★ More 6800 material than all the others combined:

MAGAZINE COMPARISON
(2 years)
Monthly Averages

	6800 Articles			TOTAL
KB	BYTE	CC	DOBB'S	PAGES
7.8	6.4	2.7	2.2	19.1 ea. mo.

Average cost for all four each month: $6.53
(Based on advertised 1-year subscription price)

'68' cost per month: $2.04

That's Right! Much, Much More
for About
1/3 the Cost!

OK, PLEASE ENTER MY SUBSCRIPTION

Bill My: Master Charge □ — VISA □

Card #________________ Exp. Date____________

For □ 1-Year □ 2 Years □ 3 Years

Enclosed: $____________

Name____________________________

Stree

City______________State__________Zip______

My Computer Is:________________________

68 Micro Journal
5900 Cassandra Smith Rd.
Hixson, TN 37343

SUBSCRIPTION RATES

USA
1 Year $24.50, 2 Year $42.50, 3 Year $64.50

*FOREIGN SURFACE Add $12.00 per Year to USA Price

*FOREIGN AIRMAIL Add $36.00 per Year to USA Price

**CANADA & MEXICO Add $5.50 per Year to USA Price
Cash (USA) or drawn on a USA Bank!!!

STAR-DOS LEVEL I

Whenever a new DOS is introduced, there's always the problem of developing software to work with it. So we did it the opposite way — we analyzed the requirements of software that already exists and developed a DOS that met them... and exceeded them! The result is STAR-DOS Level I, a new DOS for 6809 systems, ideal for single-user industrial, control, and advanced hobbyist applications. This includes SS-50 systems and single-board computers from a variety of vendors.

Level I is compatible with most current 6809 hardware and software. On the hardware side, it allows up to ten floppy or Winchester drives with appropriate controllers. On the software side, it runs existing 6809 software from all the major 6809 software suppliers, including TSC, Star-Kits, Introl, and others.

Write or call for more information. STAR-KITS Software Systems Corporation, P.O. Box 209, Mt. Kisco N.Y. 10549 (914) 241-0287.

STAR-KITS

ANDERSON COMPUTER CONSULTANTS
&
Associates

Ron Anderson, respected author and columnist for 68 MICRO JOURNAL announces the **Anderson Computer Consultants & Associates**, a consulting firm dealing primarily in 68XX(X) software design. Our wide experience in designing 6809 based control systems for machine tools is now available on a consultation basis.

Our experience includes programming machine control functions, signal analysis, multi-axis servo control (CNC) and general software design and development. We have extensive experience in instrumentation and analysis of specialized software. We support all popular languages pertaining to the 6809 and other 68XX(X) processors.

If you are a manufacturer of a control or measuring package that you believe could benefit from efficient software, write or call Ron Anderson. The fact that any calculation you can do with pencil and paper, can be done much better with a microcomputer. We will be happy to review your problem and offer a modern, state-of-the-art microcomputer solution. We can do the entire job or work with your software or hardware engineers.

Anderson Computer Consultants & Associates
3540 Sturbridge Court
Ann Arbor, MI 48105

68' MICRO JOURNAL

Disk- 1 Filesort, Minicat, Minicopy, Minifms, **Lifetime, **Poetry, **Foodlist, **Diet.

Disk- 2 Diskedit w/ inst.& fixes, Prime, *Prmod, **Snoopy, **Football, **Hexpawn,**Lifetime

Disk- 3 Cbug09, Sec1, Sec2, Find, Table2, Intext, Disk-exp, *Disksave.

Disk- 4 Mailing Program, *Finddat, *Change, *Testdisk.

DISK- 5 *DISKFIX 1, *DISKFIX 2, **LETTER, **LOVESIGN, **BLACKJAK, **BOWLING.

Disk- 6 **Purchase Order, Index (Disk file indx)

Disk- 7 Linking Loader, Rload, Harkness

Disk- 8 Crtest, Lanpher (May 82)

Disk- 9 Datecopy, Diskfix9 (Aug 82)

Disk-10 Home Accounting (July 82)

Disk-11 Dissembler (June 84)

Disk-12 Modem68 (May 84)

Disk-13 *Initmf68, Testmf68, *Cleanup, *Dskalign, Help

Disk-14 *Init, *Test, *Terminal, *Find, *Diskedit, Init.Lib

Disk-15 Modem9 + Updates (Dec. 84 Gilchrist) to Modem9 (April 84 Commo)

Disk-16 Copy.Txt, Copy.Doc, Cat.Txt, Cat.Doc, Date.Txt

NOTE:

This is a reader service **ONLY!** No Warranty is offered or implied, they are **as received** by **'68' Micro Journal,** and are for reader convenience ONLY (some MAY include fixes or patches). Also 6800 and 6809 programs are mixed, as each is fairly simple (mostly) to convert to the other.

PRICE: 8" Disk **$29.95** - 5" Disk **$24.95**

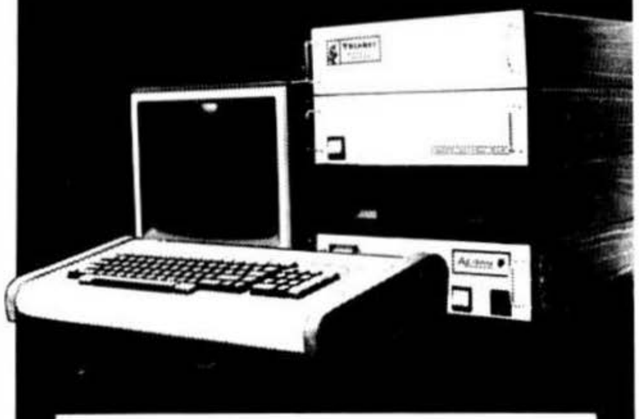
68 MICRO JOURNAL
POB 794
Hixson, TN 37343
615-842-4600 In Tenn.
1-800-338 6800 In U.S.

* Indicates 6800
** Indicates BASIC SWTPC or TSC
6809 no Indicator.

MASTER CARD - VISA Accepted
Foreign -- add **10%** for Surface
or **20%** for Air!!

ACORN™

COMPUTER SYSTEMS SS-50C



MODULES - BARE CARDS - KITS - ASSEMBLED & TESTED

Stackable Modules	KIT	A&T
20 amp POWER SUPPLY w/fan w/Disk protect relay	350.00	400.00
DISK CABINET w/regs. & cables less DRIVES	200.00	250.00
MOTHER BOARD, 8 SS-50c, 8 SS-30c NMI button	225.00	325.00

Item	Bare	KIT	A&T
IT3 - INTERRUPT TIMER 1, 10, 100 per sec.	19.95	29.95	39.95
PB4 - INTELLIGENT PORT BUFFER Single board comput.	39.95	114.95	139.95
DPIA - Dual PIA parallel port, 4 buffered I/Os	24.95	69.95	69.95
XADR - Extended Addressing BAUD gen. PIA port	29.95	69.95	89.95
MB8 - MOTHER BOARD SS-50c w/BAUD gen.	64.95	149.95	199.95
P168 - 168K PROM DISK 21, 2764 EPROMs	39.95	79.95	109.95
FD68 - Firmware development 2, 8K blocks	39.95	64.95	114.95
XMPR - 2764 PROM burner adapt. for 2716 BURNER		19.95	------
CHERRY Keyboard w/Cabinet 96 key capacitive		249.95	------
TAXAN 12", 18 Mhz MONITOR GREEN		------	149.95
AMBER		------	159.95
4 MODULE CABINET - unfinished		150.00	------
POWER SUPPLY w/disk prote t		250.00	------

Color Computer

MONOLINK - 20 Mhz Monochrome video driver	15.00	20.00
CC30 PORT BUS w/power supply 5 SS-30, 2 Cart	169.95	199.95
POWER BOX 6 switched outlets transient suppression	29.95	39.95
RS-232 3-switched ports for above	ADD +20.00	+25.00

VISA

Write for FREE Catalog
ADD $3.00 S&H PER ORDER
WIS. ADD 5% SALES TAX

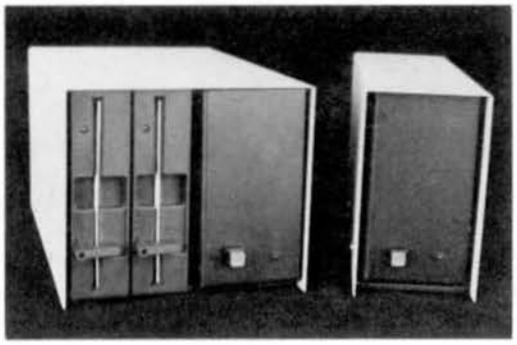
ACS INC

11931 W. Bluemound Road
MILWAUKEE, WIS. 53226
(414) 257-0300

68' MICRO JOURNAL ADVERTISERS INDEX

This index is provided as a reader service. The publisher does not assume any liability for omissions or errors.

PT69 SINGLE BOARD COMPUTER SYSTEM
OS-9 NOW AVAILABLE



Pictured:
System with Drives/System without Drives

The proven PT69 Single Board Computer now features OS-9 capability. Powerful performance, reliability, + OS-9 — UNBEATABLE! The PT69 is a complete system in a compact package.

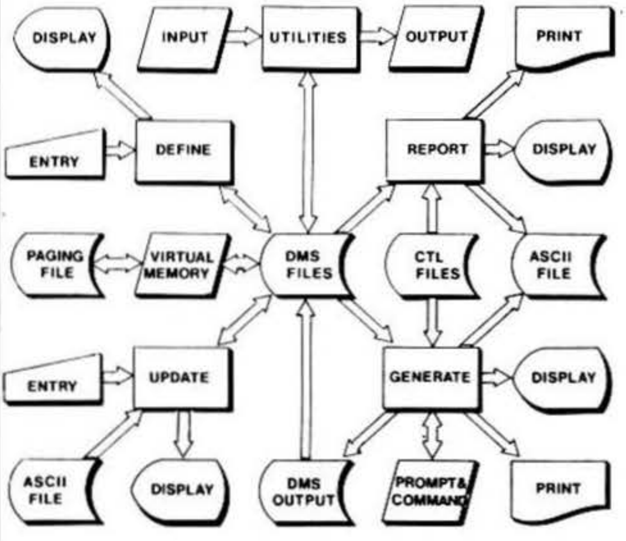
- 1 MHZ 6809E Processor
- 2 RS232 Serial Ports (6850)
- 2 8-Bit Parallel Ports (6821)
- 56K RAM; 4K EPROM
- Time-of-Day Clock (MC146818)

* COMPLETE SYSTEM with PT69 Board, 2 DS/DD 5¼" 40 Track Drives, Cabinet, and Power Supply	$999.95
* PT 69 Board, Assembled and Tested, with Power Supply + Cabinet	$399.95
* PT69, Assembled and Tested Board	$299.95
* Parallel Printer interface with cables	$ 49.95
* OS-9 L1, includes edit, asm. + debug	$250.00
* STAR-DOS Level 1 (Compatible with Flex)	$ 75.00

PERIPHERAL TECHNOLOGY
"Supplying Your Computer Needs Since 1978"
3670 Lower Roswell Road
Marietta, Georgia 30067
VISA/MASTERCARD/CHECK/COD 404/973-0042

™OS-9 is a trademark of Microware and Motorola
™FLEX is a trademark of Technical Systems Consultants

XDMS
Data Management System



System Architecture

WESTCHESTER Applied Business Systems
Post Office Box 187
Briarcliff Manor, N.Y. 10510

XDMS Data Management System

The XDMS Data Management System is available in three levels. Each level includes the XDMS nucleus, VMGEN utility and System Documentation for level III. XDMS is one of the most powerful systems available for 6809 computers and may be used for a wide variety of applications. XDMS users are registered in our database to permit distribution of product announcements and validation of user upgrades and maintenance requests.

XDMS Level I

XDMS Level I consists of DEFINE, UPDATE and REPORT facilities. This level is intended as an "entry level" system and permits entry and reporting of data on a "tabular" basis. The REPORT facility supports record and field selection, field merge, sorting, line calculations, column totals and report titling. Control is via a English-like language which is upward compatible with level II. XDMS Level I $129.95

XDMS Level II

Level II adds to Level I the powerful GENERATE facility. This facility can be thought of as a general file processor which can produce reports, forms and form letters as well as file output which may be re-input to the facility. GENERATE may be used in complex processing applications and is controlled by a English-like command language which encompasses that used by Level I. XDMS Level II $100.95

XDMS Level III

Level III includes all of level II plus a set of useful DMS Utilities. These utilities are designed to aid in the development and maintenance of user applications and permit modification of XDMS system parameters, input and output of XDMS files, display and modification of file format, graphic display of numerical data and other functions. Level III is intended for advanced XDMS users. XDMS Level III $208.95
XDMS System Documentation only ($10. credit toward purchase). . . $ 24.95

XACC Accounting System

The XACC General Accounting System is designed for small business environments of up to 10,000 accounts and inventory items. The system integrates accounting functions and inventory plus the general ledger, accounts receivable and payable functions normally sold separately in other systems. Features user defined accounts, products (or services), transactions, invoicing, etc. Easily configured to most environments.
XACC General Accounting System (Requires XDMS, pref. Lv. III). . $299.95
XACC System Documentation only ($10. credit toward purchase). . . $ 24.95

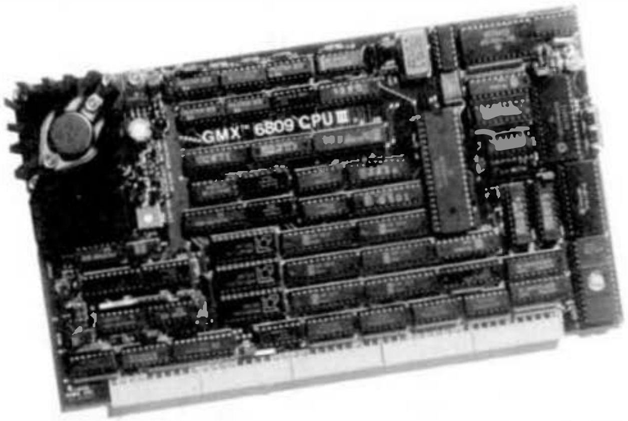
WESTCHESTER Applied Business Systems
Post Office Box 187, Briarcliff Manor, N.Y. 10510

All software is written in macro/assembler and runs under 6809 FLEX O/S.
Terms: Check, Money Order, Visa or Mastercharge. Shipment first class.
Add P&H $2.50 ($7.50 Foreign). NY Res add sales tax. Specify 5" or 8".

Sales: S. E. MEDIA, 1-800-338-6800, Consultation: 914-941-3552 (evens).

FLEX is a trademark of Technical Systems Consultants, Inc

GIMIX STATE OF THE ART 6809 SYSTEMS FOR THE SERIOUS USER.



GIMIX has 19MB or high performance 47MB Winchester Drive Systems and/or Floppy Disk Drive Systems.

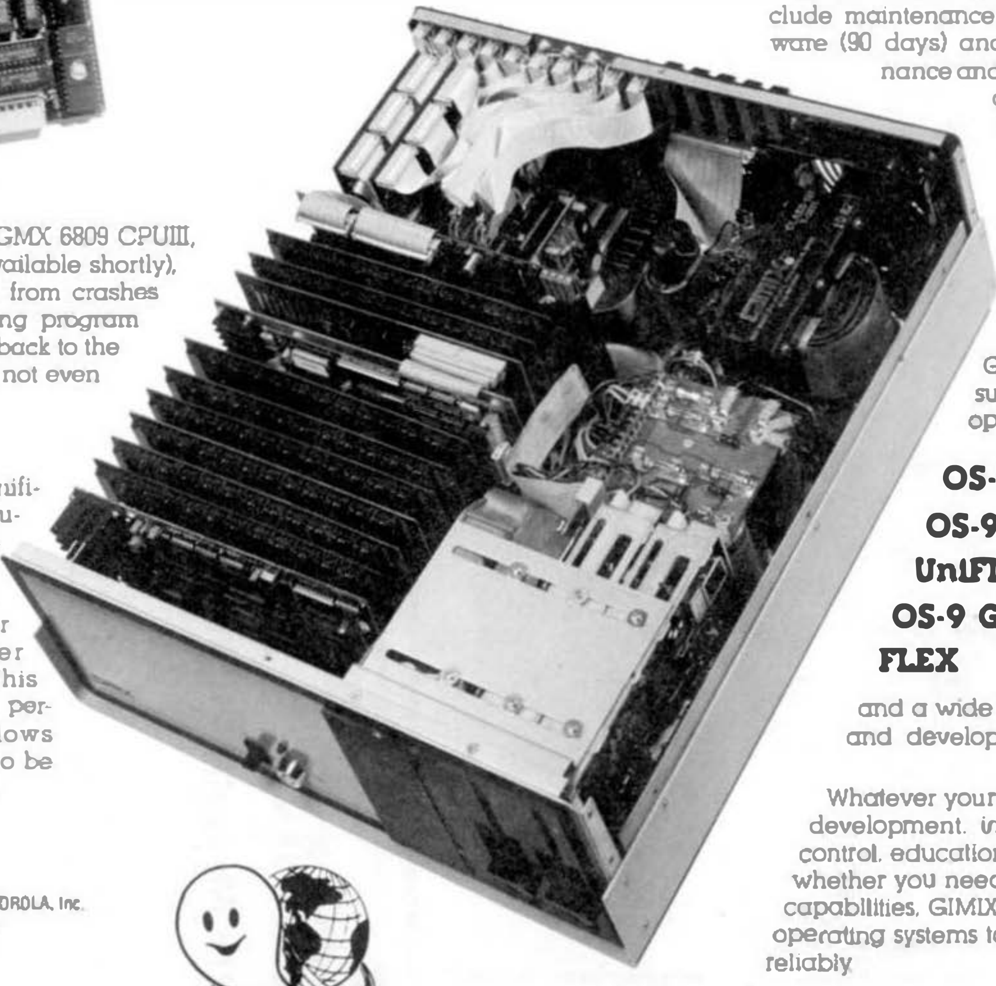
For the user who appreciates the need for a bus structured system using STATIC RAM and powered by a ferro resonant constant voltage transformer.

GIMIX has single user systems that can run both FLEX and OS-9 or Multi user systems for use with UniFLEX or OS-9.

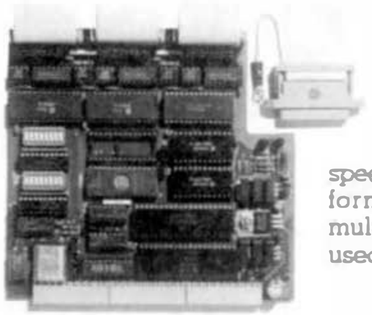
GIMIX versions of OS9 and UniFLEX include maintenance and support by Microware (90 days) and TSC (1 year). Maintenance and support after this period are available at extra cost.

(NOTE: this support and maintenance is only for use with approved GIMIX hardware)

For the ultimate in performance, the Unique GMX 6809 CPUIII, using either OS-9-GMXIII or UniFLEX GMXIII (available shortly), gives protection to the system and other users from crashes caused by defective user programs. e.g. During program development, a programmer who crashes goes back to the shell or the debugger, while the other users are not even aware anything occurred.



The intelligent serial I/O processor boards significantly reduce system overhead by handling routine I/O functions, thereby freeing up the host CPU for running user programs. This speeds up system performance and allows multiple terminals to be used at 19.2K baud.



GIMIX 6809 systems support five predominant operating systems:

OS-9 GMX III,
OS-9 GMX II,
UniFLEX,
OS-9 GMX I,
FLEX

and a wide variety of languages and development software.

Whatever your application: software development, instrumentation, process control, educational, scientific or business, whether you need single or multi-user capabilities, GIMIX has hardware and the operating systems to get the job done reliably.

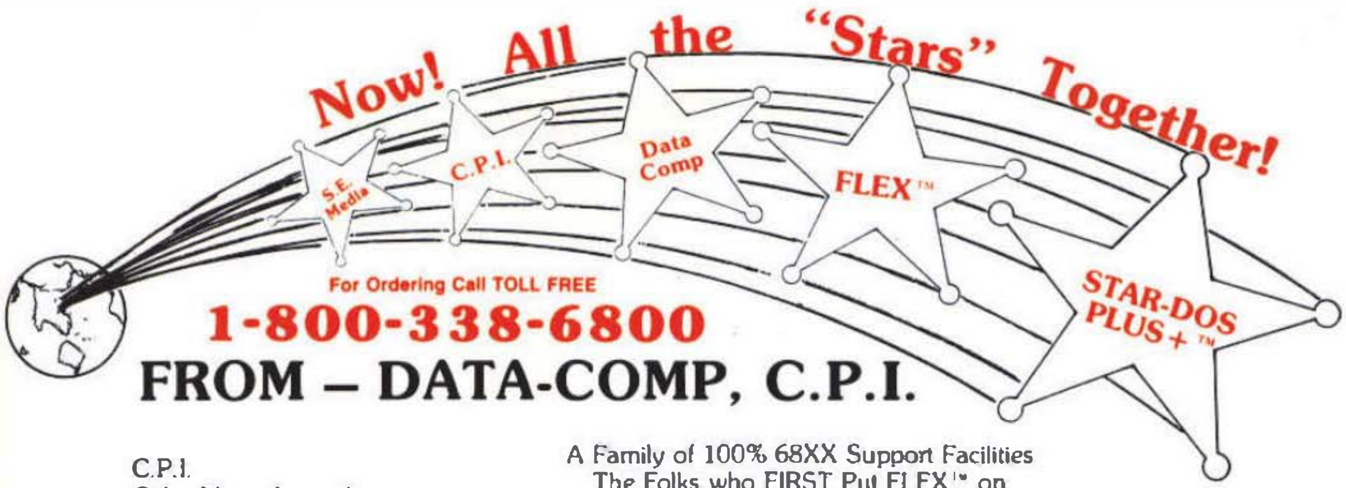
Please phone or write if you need further information.

BASIC-09 and OS-9 are trademarks of Microware Systems Corp. and MOTOROLA, Inc.
FLEX and UniFLEX are trademarks of Technical Systems Consultants, Inc.
GIMIX, GHOST, GMX, CLASSY CHASSIS, are trademarks of GIMIX, Inc.

GIMIX inc.

1337 WEST 37th PLACE • CHICAGO, ILLINOIS 60609 • (312) 927-5510 • TWX 910-221-4055

© 1983 GIMIX Inc.



For Ordering Call TOLL FREE

1-800-338-6800

FROM – DATA-COMP, C.P.I.

C.P.I.
Color Micro Journal
'68' Micro Journal
Data-Comp
S.E. Media

A Family of 100% 68XX Support Facilities
The Folks who FIRST Put FLEX™ on
The CoCo
Now Offering: *FLEX™ (2 Versions)
AND *STAR-DOS PLUS+™

FLEX-CoCo Sr.
with TSC Editor
TSC Assembler
Complete with Manuals
Reg. $250.00
Only $79.95

STAR-DOS PLUS+
- Functions Same as FLEX
- Reads - writes FLEX Disks
- Run FLEX Programs
- Just type: Run "STAR-DOS"
- Over 300 utilities & programs to choose from.

$34.50

FLEX-CoCo Jr.
without TSC
Editor & Assembler
$49.95

PLUS

ALL VERSIONS OF FLEX & STAR-DOS* INCLUDE

+ Read-Write-Dir RS Disk
+ Run RS Basic from Both
+ More Free Utilities
+ Super 800 Support
+ Free Color Micro Journal 1 yr. sub.
+ External Terminal Program
+ Test Disk Program
+ Disk Examine & Repair Program
+ Memory Examine Program
+ Many Many More!!!

TSC Editor
Reg $50.00
NOW $35.00

TSC Assembler
Reg $50.00
NOW $35.00

DISK SYSTEMS FOR THE COLOR COMPUTER

THESE PACKAGES INCLUDE DRIVE, *CONTROLLER, POWER SUPPLY & CABINET, CABLE, AND MANUAL.

* SPECIFY WHAT CONTROLLER YOU WANT J&M. OR RADIO SHACK.

PAK #1 - 1 SINGLE SIDED, DOUBLE DENSITY SYS.	$389.95
PAK #2 - 2 SINGLE SIDED, DOUBLE DENSITY SYS.	$639.95
PAK #3 - 1 DOUBLE SIDED, DOUBLE DENSITY SYS.	$439.95
PAK #4 - 2 DOUBLE SIDED, DOUBLE DENSITY SYS.	$699.95
PAK #5 - 2 DOUBLE SIDED, DOUBLE DENSITY SYS. THINLINE DRIVES, HALF SIZE	$659.95
COLOR COMPUTER II 64K W/EXT. BASIC	$189.95

CONTROLLERS

J&M DISK CONTROLLER W/ JDOS OR RADIO SHACK DISK BASIC, SPECIFY WHAT DISK BASIC.	$139.95
RADIO SHACK DISK CONTROLLER 1.1	$134.95

DISK DRIVE CABLES

CABLE FOR ONE DRIVE	$ 19.95
CABLE FOR TWO DRIVES	$ 24.95

MISC

64K UPGRADE W/MOD. INSTRUCTIONS, C.D.E.F. AND COCO 2	$ 49.95
HJL KEYBOARDS	$ 69.95
MICRO TECH LOWER CASE ROM ADAPTER	$ 74.95
RADIO SHACK BASIC 1.2	$ 29.95
RADIO SHACK DISK BASIC 1.1	$ 29.95
RADIO SHACK EXT. BASIC	$ 39.95
SCREEN CLEAN CLEARS UP VIDEO DISTORTION	$ 39.95
MEMOREX DISKS 5" SS,DD	$ 24.00
SHIPPING INCLUDED ON DISK PRICES	
DISK DRIVE CABINET & POWER SUPPLY	$ 49.95
SINGLE SIDED, DOUBLE DENSITY 5" DISK DRIVE	$199.95
DOUBLE SIDED, DOUBLE DENSITY 5" DISK DRIVE	$249.95

PRINTERS

EPSON RX-80	$325.00
EPSON RX-80FT	$375.00
EPSON MX-100	$650.00
EPSON FX-100	$799.00
EPSON FX-80	$549.00
EPSON MX-70	$200.00

SERIAL BOARDS FOR PRINTERS

MX-SERIES	$119.95
FX-SERIES	$ 99.95

USA ADD 2% SHIPPING
FOREIGN ADD 5% SHIPPING

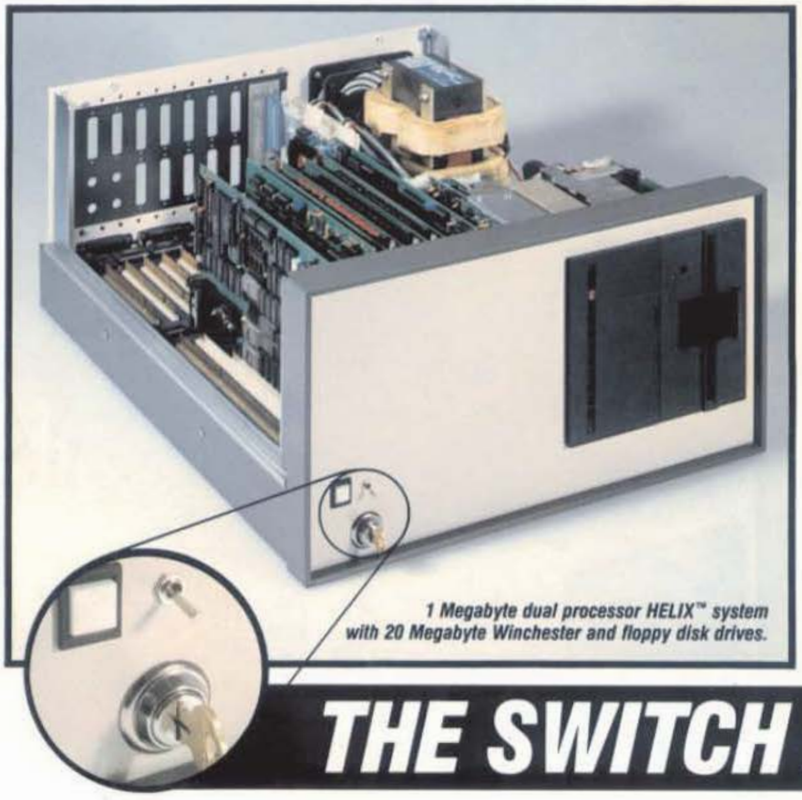
SPECIAL MX-100 $550.00

5900 Cassandra Smith Rd. Hixson, TN 37343

*FLEX is a Trademark of Technical System Consultants
*STAR-DOS+ is a Trademark of STAR-Kits & Data-Comp

68 Micro Journal
ISSN 0194-5025
5900 Cassandra Smith Rd.
Hixson, TN 37343

Second Class Postage Paid
At Hixson, TN
And Additional Mailing Offices



1 Megabyte dual processor HELIX™ system with 20 Megabyte Winchester and floppy disk drives.

ONCE AGAIN HAZELWOOD COMPUTER SYSTEMS demonstrates its leadership in computer technology by delivering the only computer system capable of switching between either the 6809 or the 68000 processor. Switching is easily accomplished by a simple front panel toggle switch. The reason we can offer this exclusive feature now, is that when our proven 6809 processor board was designed several years ago, we had the foresight to include the bus controls that allow processor switching.

Hazelwood Computer Systems is also proud to be the first S-50/S-64 bus manufacturer to license and deliver the OS9/68K Operating System from Microware Systems Corporation. OS9/68K is the 68000 version of the populer and powerful OS9 Operating System. Utilizing our proven MC-20 disk controller, OS9/68K can conveniently share a Winchester disk with OS9. Changing from 6809 to 68000 operation is as simple as switching processors and booting the new system from the Winchester disk.

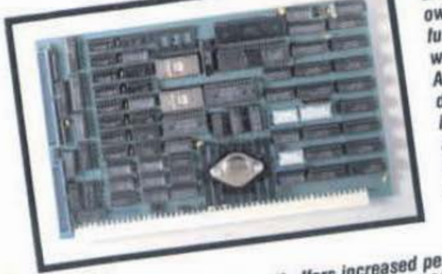
The ease of switching processors and operating systems makes a HELIX™ dual processor system the natural choice for software development. In addition, the advanced design of HELIX™ equipment, emphasizing performance and reliability, makes HELIX™ boards and systems the best value in computing offered anywhere.

System prices vary with configuration. Call for exact pricing.

THE SWITCH IS ON...

The CP-08 processor board utilizes a 68008 processor running at 10 Mhz clock rate. Using proprietary bus synchronization circuitry and single cycle DMA, the CP-08 achieves a marked performance increase over a 2 MHz 6809. Offering absolute compatibility with the 68000 instruction set, the 68008 addresses up to 1 Megabyte of memory. Also included on the CP-08 are up to 4K of ROM, an interrupt timer, and, with battery backup operation, a clock/calendar and 2K RAM. Implemented as a standard S-50 board, the CP-08 brings 68000 operation to S-50 bus computers.

ORDER: CP-08 PRICE: $595



The MC-20 Mass Storage Controller board interfaces up to 4 floppy and 8 Winchester disk drives to the S-50/S-64 bus. The MC-20 is an intelligent controller with its own 2 Mhz 6809 processor and 56K RAM. It provides DMA data transfers to a full 24 bit address. All disk operation requests are by logical block number, with the controller performing the necessary track/sector address calculations. Any combination of 5¼ or 8 inch floppy drives can be accommodated with all drive parameters, such as write precompensation, software controlled for each individual drive. Winchester drives are connected via a SASI bus interface. Block address mapping is provided which allows a single drive to be segmented into several logical units. The MC-20 is the controller of the MS-20 Mass Storage Subsystem which includes a 20 Megabyte Winchester drive.

ORDER: MC-20 PRICE: $695

OS9/68K

OS9/68K offers increased performance and larger user memory space while retaining all of the features of OS9. Disk file compatibility and operational similarity assures that present OS9 users can easily transfer their operations to the 68000. Included are an editor, assembler, linker, and debugger. A C compiler is available now. BASIC09 and other languages will be avilable soon.

ORDER: OS9/68K PRICE: $250

All items available stock to 30 days.
Prices subject to change without notice.

HAZELWOOD COMPUTER SYSTEMS

907 East Terra, O'Fallon, MO 63366, 314-281-1055

HELIX™

OS9 and OS9/68K are registered trademarks of microware Systems Corp. HELIX is a trademark of Hazelwood Computer Systems.